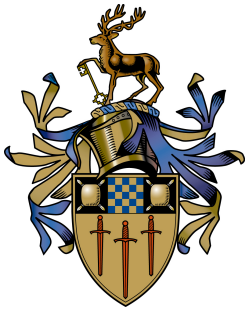
Active Sampling for Computer Vision

Yusuf Duman

Submitted for the Degree of
Doctor of Philosophy
from the
University of Surrey



Centre for Vision, Speech and Signal Processing
Faculty of Engineering and Physical Sciences
University of Surrey
Guildford, Surrey GU2 7XH, U.K.

September 2022

Abstract

Mammalian vision systems do not view an entire scene in one go. Instead, rapid eye movements known as saccades point the high density areas of photoreceptors in the retina toward areas of detail. Consequently, a detailed view of the scene can be built by the brain using a relatively small amount of information. By integrating the imaging in this manner the quality of the visual processing found deeper within the brain is improved as it only has to process the salient details.

A scanning pixel camera presents a way of realising this in hardware. A low cost, low power sensor system that builds up an image of a scene by rapidly sampling a sensor that sits behind a moveable set of optics. Advances in micro-actuation allows the low-cost optics to be scanned across the scene in a programmable manner. This can lead to the lens-less zooming effects by simply varying the scan speed or the sample rate. Furthermore, the amount of information that this type of sensor provides can be varied by simply changing the scan pattern.

However, a major drawback of this type of sensor system is that it takes a long time to image a full scene when compared to a traditional CCD camera. This motivates the work of this thesis to find a scan pattern that allows the best use of the saccade-like behaviour of a scanning pixel camera. By focusing on scene details relevant to a predefined computer vision task, this thesis demonstrates that it is possible to produce a scan pattern that allows us to overcome this major issue. In this thesis we provide methods of generating useful sample maps that enhance the abilities of a scanning pixel camera and make it an efficient part of a computer vision pipeline.

By actively providing sample patterns to the scanning pixel camera, the sensor becomes an active part of the computer vision system, rather than simply a source of data. This is similar to the purpose of saccades in a mammalian vision system. In doing this we create another challenge that is addressed in this thesis. Namely, the downstream computer vision task has only a partial view of the scene, that may be affected by different types of artefacting found in scanning pixel cameras. Therefore, how do these tasks need to be adapted to deal with data in this form, both during training and inference.

This thesis approaches this problem by first making several assumptions about a scanning pixel camera to adapt existing computer vision techniques to find useful sample patterns. These initial assumptions include that scene is static and is imaged with full knowledge of its contents. These are then used to create simple model of an scanning pixel camera to establish the best possible way of generating sampling positions for a downstream task. These assumptions are then progressively removed in order to finally reach a method that can be deployed on a real system. The end result is a technique that requires no prior knowledge of the scene to begin with, forcing the scanning pixel camera to explore the scene before it knows what it is looking at.

The sample maps generated are designed to generate images to be used by a downstream computer vision, rather than viewed by a human. To evaluate this we apply this technique to a variety of computer vision tasks and demonstrate that such a piece of hardware can form a useful part of a computer vision system. These tasks include object classification, tracking and instance segmentation.

Acknowledgements

I would first like to thank my supervisors, Dr. Simon Hadfield and Dr. Jean-Yves Guillemaut, for providing invalauble support and assistance in completing this PhD. Particularly for discussing ideas with me, helping me out when I became stuck on a problem, and for spending vast amounts of time to proofread my writing. I would also like to thank my industrial sponsor Barry Clive of Nimrod Systems Ltd. for providing the funding and initial concept necessary to allow me to have the opportunity to undertake this PhD.

Further thanks goes to my family who have always supported me from the very beginning and have enabled me to get this far. Also thanks to everyone at CVSSP who helped me and made this a great place to work. Particular mentions go to Tommy, who has been with me for the whole journey, especially during lunch. Violeta, for providing food and moral support, particularly over the pandemic. Lucy, Celyn, Jaime and everyone else I have shared an office with, for providing useful discussion and helping with the finer technical details.

Finally, I would like to thank Gamesoc and everyone at Surrey Taekwondo for providing me with an outlet unrelated to my PhD and helping me stay sane during this entire four year process.

Declaration

This thesis and the work to which it refers are the results of my own efforts. Any ideas, data, images or text resulting from the work of others (whether published or unpublished) are fully identified as such within the work and attributed to their originator in the text, bibliography or in footnotes. This thesis has not been submitted in whole or in part for any other academic degree or professional qualification. I agree that the University has the right to submit my work to the plagiarism detection service TurnitinUK for originality checks. Whether or not drafts have been so-assessed, the University reserves the right to require an electronic version of the final document (as submitted) for assessment as above.

Signed: Yusuf Duman

Date: September 10, 2022

Contents

Nomenclature

BMVC British Machine Vision Conference

ICCV International Conference on Computer Vision

AI Artificial Intelligence

AR Augmented Reality

SDK Software Development Kit

FPS Frames Per Second

DMD Digital Micromirror Device

SSIM Structural Similarity

GMLC Generalized Maximum Likelihood Classification

mAP Mean average precision

GNN Graph Neural Network

FOV field-of-view

CNN convolutional neural network

MAR mixed adaptive-random sampling

FPN Feature Pyramind Network

Symbols

Notation Conventions

x	Scalar values are lowercase
$\mathbf{x}$	Vector values are lowercase bold
$\mathbf{X}$	Uppercase bold values represent a tensor of size $C \times H \times W$
$\mathcal{F}$	Calligraphic font represents functions
t	t is a scalar value used to represent time
$\mathbf{X}^q$	Superscripts are used to index members of set with common properties
$\mathbf{X}_t$	Subscript t is used to denote an item that can change value at each time step

Introduced in Introduction

κ	sample rate
θ	angular resolution
$\dot{\theta}$	angular velocity
$\mathbf{S}$	sample map
$\mathbf{I}$	image
$\hat{\mathbf{I}}$	scanning pixel camera image
α	acceptance angle
N	maximum number of samples that can be taken from a scene
$\mathbf{r}$	image resolution
p	scan overlap
$\mathbf{K}$	overlap kernel

Introduced in the Literature Review

$\mathbf{D}$	binary matrix with each row representing a Digital Micromirror Device configuration
$\mathbf{y}$	compressed sensing measurements
$\mathbf{i}$	vector representing the image

Introduced in Chapter 1

n	number of samples taken from a scene
i	sampling position
$\mathbf{s}$	a sample value
$\mathcal{G}$	sampling operation
$\mathcal{S}$	heatmap generator
$\mathcal{B}$	heatmap binariser
κ_a	target sample rate
$\mathcal{T}$	downstream computer vision task
$\mathcal{M}$	metric for a downstream computer vision task
$\mathbf{H}$	sample heatmap
$\mathbf{d}$	sample distance
$\mathbb{I}$	set of all possible samples
$\mathcal{E}$	feature encoder module
$\mathcal{D}$	feature decoder module
$\mathbf{F}$	intermediate feature representation
$\mathbf{y}$	task ground truth

Introduced in chapter 2

$\mathbf{w}$ weights

$\mathbf{G}$ task related ground truth mask

$\mathcal{D}$ scheduling function

x_t temporal component

$\mathcal{R}$ residual block

Introduced in chapter 3

$\mathcal{S}$ saliency predictor network

MLS maximum logit score

$\hat{\mathbf{y}}$ task prediction

$\mathcal{O}$ oracle system

ψ downsampling scaling factor

List of Figures

List of Tables

Chapter 1

Introduction

Modern specialist optics seek to optimise the transfer of light between source and target. It is relatively simple to create an optical system which only accepts light within a certain acceptance angle (α). An example of the design of these optics can be seen in fig. 1.1. If such optics are used to transfer the light to a single photodetector placed behind the optical system, it will collect a large amount of light from a specific chosen direction, giving a stronger signal from within the direction covered by α. This reduces the effect of sensor noise and allows measurements to be taken more rapidly in in a wider variety of lighting conditions.

Taking a single measurement while looking in a fixed direction does not provide much useful imaging data, being akin to a single pixel in an image. However, by moving the optics so they point in a different direction and repeatedly sampling the sensor as this is done, an image can be built up over a period of time. This technique is to produce images has been used with more traditional cameras in earth observation satellites, such as Landsat 7, and has been referred to as a whisk broom scanner [1]. In this thesis, the entire system, both the optics and the photodetector behind them will be referred to as a scanning pixel camera, as an image is being built using as a pixel sensor is scanned across a scene. An example of such a camera and how it might image a scene is shown in fig. 1.2.

To image a scene the optics are moved across the scene and the receiver is repeatedly sampled at a frequency, κ. Each sample will correspond with a different, and potentially overlapping, spatial location. We can treat these samples as pixels and by arranging them according to their spatial location a viewable pseudo-image is formed, as shown in fig. 1.3.

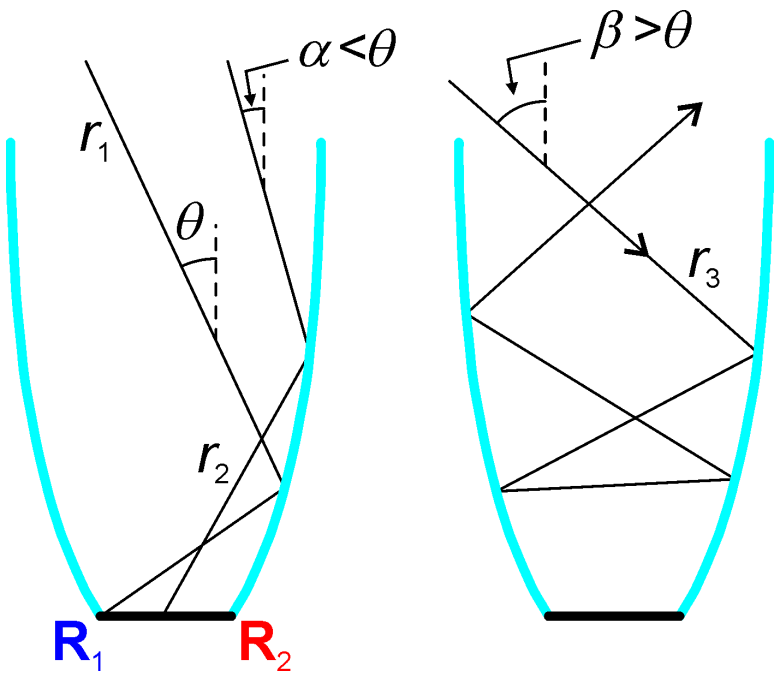
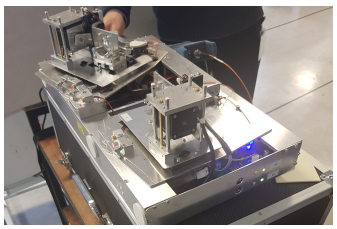


Figure 1.1: Optics, showing the acceptance angle (α) [66]. Many designs are possible but all will attempt to maximise the transfer of light toward $\mathbf{R}_1$ and $\mathbf{R}_2$ as long as the incident rays come from within α as shown in the left diagram. Light rays from outside this angle (β), are rejected as shown in the right diagram.



(a) A scanning pixel camera

(b) How a conventional camera would image a scene

(c) How scanning pixel camera might intelligently image a scene

Figure 1.2: A single pixel camera and an example of how it might optimally sample a scene in order to retain only the relevant detail

The unique operation of the scanning pixel camera can utilised in a variety of ways. As the optics maximise the amount of light from the source, the scanning pixel camera is effective in low light conditions, and the improvements in the signal-to-noise ratio allow for more precise measurements of the incident light. Also, as there is only a single photodetector, the scanning pixel camera will be useful in low power environments.

A scanning pixel camera can theoretically achieve very high resolutions. The acceptance angle of the optics places a practical limit on the angular separation of any two points. As long as the acceptance angle is sufficiently small, a high sample rate can be used to gather a large amount of samples from a scene. Notably, the resulting resolution of the image can be adapted on the fly simply by changing the sample rate. This could even occur during imaging, resulting in the density of pixels in the pseudo-image potentially varying in different locations depending on the sampling pattern used. If the scan pattern (*i.e.* the path of the moveable optics) is kept fixed, and the sample rate is increased or decreased, there is a respective increase or decrease in the resolution of the recovered pseudo-image, within the same field-of-view.

It is also possible to change the scan pattern to achieve different optical effects. If the same sample rate is maintained and the scan is performed over a smaller area zoom effects can be achieved without the need for lenses, enabling the device to take a smaller form factor. The resolution of the pseudo-image will be maintained, but as it views a smaller portion of the scene, it will appear zoomed in.

Although the scanning pixel camera offers many advantages, there are some drawbacks of using one. If there is an overlap in spatial location in which the light is gathered, the resulting

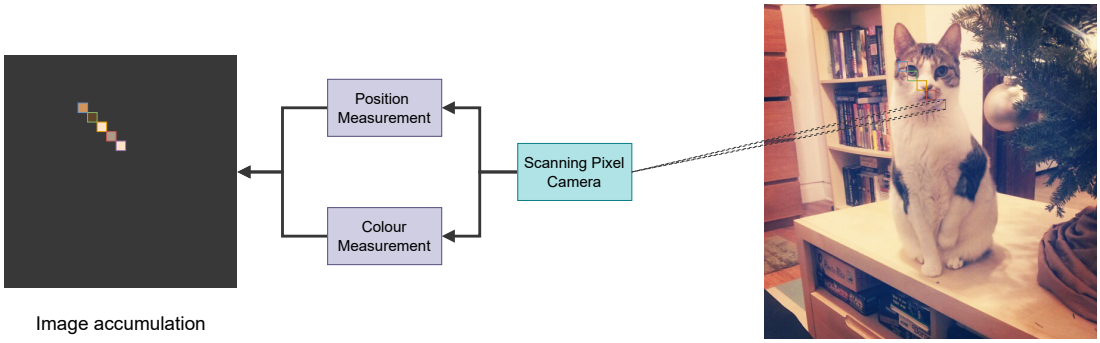


Figure 1.3: The operation of a scanning pixel camera. As the scan moves along a predetermined scan path, the pixels are sampled and accumulated into an scanning pixel camera image.

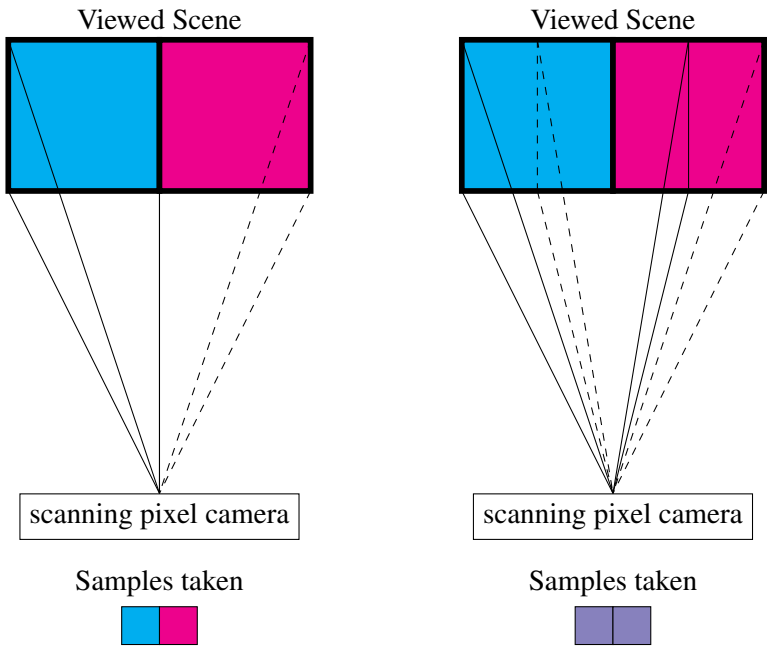
pseudo-image will appear blurred when compared to that of a traditional RGB camera, as demonstrated in fig. 1.4. This is because the aggregation of light by the optics results in a blur if the areas viewed overlaps between samples. This will occur when taking samples that are less than α° apart. Therefore, while very high resolution is possible, in order to do it without having information lost to blurring, the following constraint between sample rate (κ), angular velocity ($\dot{\theta}$) and α applies,

$$\kappa \leq \frac{\dot{\theta}}{\alpha} \tag{1.1}$$

For the purposes of this thesis, it is helpful to restate this in terms of the field-of-view (FOV) of the scanning pixel camera, *i.e*. the maximum possible area that can be scanned,

$$\kappa \leq \frac{\text{FOV}}{\alpha} \tag{1.2}$$

Although the sensor can theoretically achieve extremely high resolutions, eq. (1.2) shows that limiting factor of the scanning pixel camera is the time taken to acquire an image, particularly if a high sample rate is being used. Furthermore, large amounts of data will make it expensive to transmit and process the image, which somewhat negates its advantages, particularly in a low power setting. Therefore there is an incentive to minimise the number of samples taken while maintaining the quality of the image. This is possible due as the scan pattern can be freely manipulated to only sample part of the scene. This will speed up image acquisition, while reducing processing and transmission times, and reduce any artefacts produced by the scene changing over time (*e.g*. rolling shutter artefacts). Ideally the scanning pixel camera should



(a) No scan overlap

(b) With Scan Overlap

Figure 1.4: Figure 1.4a shows how scanning with no overlap is able to resolve features of the scene that are at least α apart. If the sample rate is too high for the given α, a blurring of the samples occurs, as shown in fig. 1.4b.

reduce the sample rate for areas that require a lower resolution as it will improve the time taken to form an image without a loss of information.

This idea can be taken further, discarding not only samples which contain no information, but also discarding samples whose information is not valuable for a particular task. While this may not be of use if the desired effect is to get a complete image of the scene, it may be useful if the resulting pseudo-image is being used in an automated computer vision pipeline. In this case the scanning pixel camera may only need to scan the positions that are relevant to the computer vision task. What makes a particular sample relevant will depend on how the data from the scanning pixel camera is going to be used. For example, if it is known that the camera is going to be deployed in a fixed location, with the intent on looking directly at something, then the sampling pattern can be used to ignore areas of the scene that are not necessary. Alternatively, if something is know about the task that the data is used for then the sampling can be optimised to accomplish this task. Determining which samples these should be however, requires a computer vision based solution.

1.1 Non-Traditional cameras

The scanning pixel camera is one of many designs of camera that can be considered non-traditional when compared to a common RGB CCD based camera. The simplest of these would be a hyper-spectral camera, where the CCD responds to electromagnetic frequencies beyond the visible range, such as infra-red cameras. These types of sensor are particularly useful in remote sensing as they give information that is normally not visible. While this data may be useful for many applications, the lack of labelled data in these domains limit it's use in a computer vision context.

Other designs of camera, such as event cameras, which use active pixel sensors to provide asynchronous information on brightness changes in the scene, or compressed sensing based imaging, which uses similar hardware but aims to reconstruct the an image while sampling below the Nyquist frequency have also been developed. The existence of such sensors and their applications demonstrates that there is a need for non-traditional cameras as traditional cameras have limitations that prevent them from being deployed in all scenarios.

These cameras do not provide data in an imaged based format. This makes it difficult to adapt existing computer vision algorithms to use these cameras, preventing their wider uptake. The same is true for a scanning pixel camera, as both existing algorithms need to be retrained to understand the pseudo-images, and a scan pattern for a particular computer vision task needs to be derived to determine which samples will be most relevant. Despite the potential utility of using non-traditional cameras, little work has been done to adapt existing computer vision methods to exploit the unique advantages of these cameras. This thesis will adapt existing computer vision algorithms so that they can be used by scanning pixel cameras.

1.2 Applying computer vision to scanning pixel cameras

The scan pattern of a scanning pixel camera being completely controllable is highly valuable for numerous computer vision problems. The biggest challenge when deploying deep learning solutions to real-time problems such as tracking is the sheer quantity of data to process. Recent advances in parallel processing architectures have alleviated this but they require expensive, power-hungry hardware. The volume of data is so high because vision is an extremely redundant sensing modality, despite only a tiny portion of the scene being useful for a given task.

Biological vision systems attempt to alleviate these problems in two ways. Firstly, photoreceptor cells are arranged in a non-uniform pattern, providing high detail at the centre of the view and rapidly dropping further away from this. This concept is also utilised in foveated rendering, where less rendering compute is given to areas in the viewer's periphery, allowing for faster rendering without sacrificing fidelity. Secondly, unconscious eye motion (saccades) and a complex visual processing system allows challenging tasks to be solved from a relatively small amount of information [63].

As the scanning pixel camera has the ability to be pointed in a specific direction, this can be utilised to build a system that mimics saccades in a biological vision system. Specifically, the sensor should actuate itself in order to specifically target areas which it believes contain useful information for solving the task, based on it's observations so far. A clear example of the utility of this is in a task such as tracking, where it would be useful to focus on the position of the target in future frames, and not image any distractors in the scene. This would then require the

sensor to predict where it thinks the target will be in the next frame, and take samples in that area. However, we should note that biological vision systems are not bound by frames, and are instead continuously building an image as the eyes scan over a scene. This leads to the most fundamental implication of the single pixel camera, the idea that frames may not be needed at all.

Provided that areas of the scene contain the salient information for completing a given computer vision task, this thesis will show it will be possible to complete the task given a much smaller portion of available visual information. Doing this makes the sensor an active participant in solving the computer vision task. This allows the scanning pixel camera to be effective for computer vision tasks as it is low-cost and low-power, meaning it can be widely deployed, particularly for tasks in an internet-of-things setting and for space-based remote sensing.

This thesis proposes methods of generating task specific sample patterns. Such patterns would contain the minimum possible number of samples while still maintaining task effectiveness. To do this they should mimic a biological vision system's ability to focus on salient details of a scene.

1.3 Data representation

This thesis involves the use of experimental hardware, which provides image data in a novel format, however, existing computer vision algorithms do not operate on data in this format. The data coming from a scanning pixel camera can be converted into a traditional RGB image as follows. The sensor images the scene following a pre-determined scan pattern. This provides a mapping between an input domain of potential sampling positions ($i \in \mathbb{R}^2$) and a d dimensional vector space of intensity values.

$$\mathcal{G} : \mathbb{R}^2 \mapsto \mathbb{R}^d \tag{1.3}$$

The value of d depends on the photodetectors used, however to be able to utilise the scanning pixel camera to existing computer vision methods this shall be assumed to be an RGB photodetector (*i.e.* $d = 3$). By quantising the space of sampling positions, each sample can be assigned a pixel on a grid. The resolution of this gird will be related to the other properties of the scanning pixel camera as described in eqs. (1.4) to (1.6).

There is also a lack of data required to train computer vision models. There is therefore a need to simulate scanning pixel camera data in order to effectively train the deep-learning models that modern computer vision requires. Fortunately, it is possible to use images from a traditional camera in order to generate data. This has the additional advantage of already being annotated for a wide variety of computer vision tasks. Therefore, scanning pixel camera data can be simulated by treating each pixel in an image from a standard camera as a sample taken by a scanning pixel camera. In this case the maximum field-of-view (FOV) is determined by the camera that took the original image. Given the image resolution ($\mathbf{r}$), this will also determine the acceptance angle (α), the maximum sample rate (κ), and the maximum number of samples that can be taken from a scene (N) as follows,

$$\alpha = \frac{\text{FOV}}{\mathbf{r}} \tag{1.4}$$

$$N = \mathbf{r}_x \times \mathbf{r}_y \tag{1.5}$$

$$\kappa = \frac{\mathbf{r}}{\text{FOV}} \tag{1.6}$$

A sample pattern can be represented as a binary matrix, referred to as the sample map ($\mathbf{S}$). This can be resized to fit a given image resolution, and consequently any α, maximum κ, and N that is required. Assuming that the constraint in eq. (1.2) is being followed, this means that the relationship between an image ($\mathbf{I}$) and a scanning pixel camera image ($\hat{\mathbf{I}}$) can be defined using the element-wise product $\odot$,

$$\hat{\mathbf{I}} = \mathbf{I} \odot \mathbf{S} \tag{1.7}$$

If eq. (1.2) is no longer being obeyed then a box blur must be applied to account for the summation of incident light. This can be represented by a cross-correlation with a matrix of ones ($\mathbf{1}$), the size of which depends on the amount of scan overlap (p). This defines the size of the overlap kernel ($\mathbf{K}$),

$$s = 2p + 1 \tag{1.8}$$

$$\mathbf{K} = \frac{1}{s^2}\mathbf{1}^{s \times s} \tag{1.9}$$

Thus eq. (1.7) is modified as follows, using $*$ as the cross-correlation operator,

$$\hat{\mathbf{I}} = \mathbf{K} * (\mathbf{I} \odot \mathbf{S}) \tag{1.10}$$

Note that the box blur is un-normalised if the overlap was resulting from a relative increase in the acceptance angle, as this would result in more light being received by the sensor. If the blur was cause by an increase in samplerate, then the blur would have to be normalised as the amount of light on each sample would remain the same. Additionally, $\mathbf{K}$ can be changed to reflect different shaped optics and light gathering patterns, should these be non-uniform.

The fact that a blur is used opens the door to deconvolutional techniques for de-blurring, such as Weiner deconvolution [14]. Depending on the nature of the motion and the number of samples taken, the value of $\mathbf{K}$ will change, as the overlap will take on a different shape. While these may benefit some applications, particularly where detail is required, they are not applied here. Instead we rely on the ability of a neural network to overcome these deficiencies in the image and still produce a reliable result.

This allows us to leverage the large amount of already labelled data in order to train on simulated scanning pixel camera. While the camera parameters from eqs. (1.4) to (1.6) will not be recoverable if this information is missing from the original photo, this is actually a benefit in a deep learning context as the models produced will be generalised to a wide variety of scanning pixel camera parameters. While this method is still limited by the optical properties of a traditional camera, this is useful for building modern computer vision algorithms for scanning pixel camera enabled systems. The simulation also does not account for the speed of acquisition, which will depend on the length of the path taken in order to image the scene. This will limit the effects of motion blurring as we will be restricted to whatever motion blurring was found when the original image was taken.

1.4 Aims

The aim of this thesis is to use the scanning pixel camera to bridge the gap between the sensor and the computer vision task that the resulting data is used for. This will allow the scanning pixel camera to be easily adapted to a variety of scenarios where it would be beneficial to perform such tasks with the added benefit of the properties of the scanning pixel camera. To accomplish this, this thesis will initially simplify the method by which scanning pixel camera images are generated in order to make it easier to convert existing computer vision algorithms

to the scanning pixel camera domain. By progressively removing these simplifications and adjusting the methods used to process the computer vision tasks, this thesis will eventually produce algorithms that are suited to the more complex task of operating on scanning pixel camera data.

The simplifications needed to treat a single pixel camera equivalently to a normal camera are primarily focused around the mechanism used to apply the sample map. These are as follows,

1. We have existing information on the scene we are sampling. In practice this will not be the case and we will have to dynamically build the best possible sample map based only on pre-existing samples.

2. The scene being imaged is static. However, a scene is likely to move during capture so there is likely to be motion blurring in any acquired scanning pixel camera image. There will also be rolling shutter artefacts however owing to the difference of motion with a traditional camera shutter they would manifest differently.

3. The task can only be performed on a complete frame of samples. As a scanning pixel camera is mimicking the behaviour of saccades, the concept of a frame appears arbitrary and there is no need for such a timing constraint.

As this thesis progresses these simplifications will be removed so that the algorithms developed approach a real-world use case scenario for a scanning pixel camera.

In order to achieve these aims, this thesis presents the following contributions: Firstly, in chapter 2, we will explore attentional mechanisms and sampling techniques. This will yield information on how sampling is currently performed and how input samples are evaluated based on how useful they will be. We will also take a look at alternative imaging techniques in order to understand existing methods of dealing with non-conventional image data.

In chapter 3 the best way to generate useful sampling patterns, given all the simplifications above, will be established. This will give an understanding of what a sample map that is optimised to a given task will look like, as well as any task dependant variation that may apply. Furthermore, it will be used to establish a lower bound on the number of samples that can be used to image a scene while still preserving task performance. Doing this will provide data to

which developments in future chapters can be compared against. Towards the end of this chapter, some techniques to remove simplification 1 will be explored.

Chapter 4 focuses on removing simplification 1 and simplification 2. This is done by moving to video-based computer vision problems and having the sample maps be produced one frame in advance, resulting in a predictive system that can be applied to any video based computer vision task. This will also help further establish practical lower bounds on the number of samples that can be used.

The third technical contribution in chapter 5 focuses on removing simplification 3. Utilising the knowledge gained in chapters 3 and 4 at truly predictive system for sampling with a scanning pixel camera will be proposed. This will continuously pick individual samples until a task is done, with no reference to the frame. It will be applied to a number of computer vision tasks, showing that it can benefit image acquisition time.

After having removed all the simplifications and arriving at a system that can be used in a real-world scanning pixel camera, chapter 6 summarise the overall results and presents several future direction which the field can take.

Chapter 2

Literature Review

This chapter examines the literature related to this thesis, which is split into several parts. Firstly, this involves looking at existing scanning pixel camera imaging methods using compressive sensing, as these use a similar hardware setup. Secondly, we will look at event cameras to provide some context as to why non-traditional cameras are useful, particularly in a computer vision setup. Then, visual explanations and attention mechanisms in neural networks are examined in order to motivate the idea that it is beneficial to look at certain areas of input. Finally, existing sampling techniques are surveyed in order to establish current state-of-the-art methods that the algorithms in this thesis will need to be compared against.

The literature presented in this chapter forms a general background for the thesis as a whole. In later chapters we will introduce specific computer vision tasks to apply the scanning pixel camera to. The specific literature for these tasks will be reviewed as these tasks are introduced.

2.1 Compressive Sensing Based Imaging

Compressive sensing can be used to reconstruct images with fewer samples than the Nyquist limit would require [18]. Under this setup, a single photodetector is used, similar to our scanning pixel camera. However, it is stationary and uses a Digital Micromirror Device (DMD) consisting of an array of tiny, switchable mirrors modulate the incoming light. The DMD is set to random, sparse patterns and for each pattern the value at the photodetector is measured. This process is repeated

multiple times, with different patterns. By using a binary matrix with each row representing a Digital Micromirror Device configuration ($\mathbf{D}$), the compressed sensing measurements ($\mathbf{y}$) and a vector representing the image ($\mathbf{i}$), the following under-determined linear system can be formed,

$$\mathbf{y} = \mathbf{D}\mathbf{i} \tag{2.1}$$

This can then be solved for $\mathbf{i}$ to produce the image [5]. This is possible even though the system is under-determined due to the sparse nature of $\mathbf{D}$.

Various optimisations have been made on this technique, in order to improve the speed of imaging by a factor of 1000 [6] and to allow it to record video [3]. A more modern deep learning based approach [27] uses an auto-encoder to learn a fixed set of sampling patterns for the DMD to use. The resulting measurements can then be passed into a decoder and the image can then be reconstructed. This has been used to improve the quality of the reconstruction.

Having a fixed optical system makes compressive sensing far less flexible than the approach proposed in this thesis. It is challenging or impossible to adapt camera properties like resolution and field of view on the fly. Furthermore, compressive sensing techniques tend to focus on the reconstruction of a full image based on a small number of samples. In contrast the work proposed within this thesis attempts to make the sensor an active participant in a computer vision pipeline. In order to do this, the image reconstruction step is bypassed entirely, and the samples are directly fed into a downstream pipeline. For this to be effective the samples must be well chosen for the specific computer vision task.

Some previous work has been done directly integrating a sampled image into an action recognition pipeline [33]. Here 3D MACH filters were used to condition the random patterns in a DMD array to allow for classification of the action directly from the output rather than performing a reconstruction. Also of interest here is the smashed filter [11]. This extends Generalized Maximum Likelihood Classification (GMLC) to be used with compressed sensing data for classification and target recognition. To do this the GMLC is treated as if it is being performed on an orthoprojection of the input, as defined by the value of the DMD.

Both these methods are designed for specific tasks and do not use modern computer vision techniques to perform this. While the methods proposed in this thesis also propose samplings (albeit more dynamic ones) designed to optimise the success of a downstream task, they will

leverage the commonality found in current deep learning techniques in order to make a task generic technique.

2.2 Event Cameras

As mentioned in chapter 1, there are uses for non-traditional types of cameras as traditional cameras can be limited, particularly when certain properties are required for a task. For example, in order to track objects that move at high speed, a high framerate camera is required. This will produce a lot of data which has to be processed very quickly, which places a practical limit on what it is possible to track.

To overcome this, it is possible to use event cameras [35]. These provide an asynchronous stream of data that indicates when the brightness of a pixel has changed. Owing to the asynchronous nature of the data provided, the latency can be significantly lowered to $15\,\mu s$. As the sensor has the temporal resolution to see the fast moving object, but does not place the onerous constraint of large volumes of data, it is well placed to be used for tracking in this context [21].

Many other applications, such as pose estimation [41], SLAM [42] and image reconstruction [51] are also possible. Each of these are able to leverage the event camera to produce a result in contexts where it would be difficult for a traditional camera. These demonstrate the utility of the event camera and why it is necessary to overcome the limitations of a traditional camera.

As a scanning pixel camera is low-power it can be used in remote locations where there is a limited power supply. Alternatively, it can be used in a internet-of-things context, where sensor power and bandwidth consumption should be limited due to the large amount of sensors being used. Also, as the optics will maximise the light gathered, it is useful in a low-light context, allowing it to provide data where a traditional camera may not be able to. These show that a scanning pixel camera provides it's own way of overcoming the limitations of a traditional camera. This thesis seeks to enable that by using intelligent sampling to overcome the internal limitations of a scanning pixel camera and provide useful data for computer vision tasks.

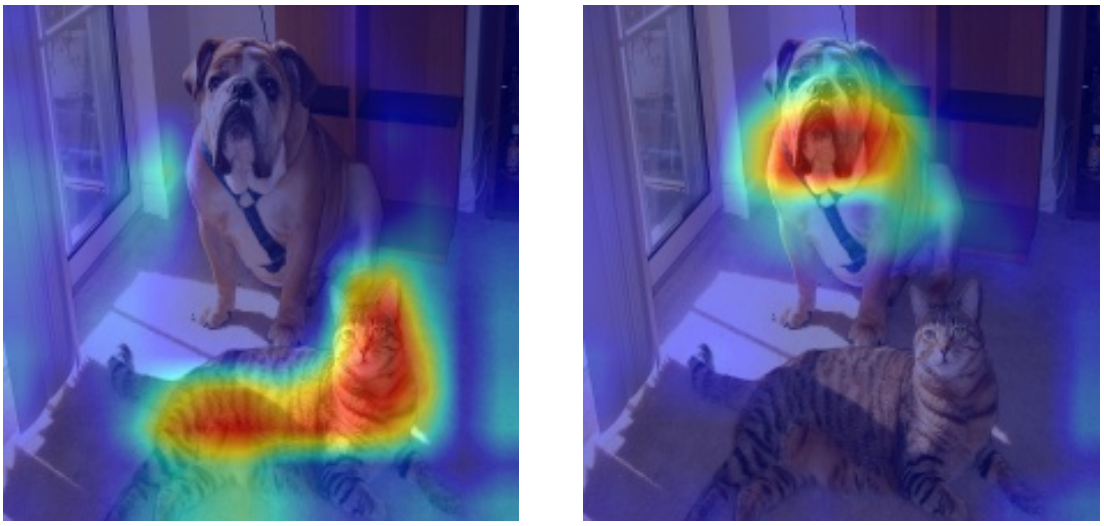


Figure 2.1: Example images produced by GradCAM [22], showing which pixels are causing the classifications for the cat and dog classes respectively. As this thesis will show, it is possible to to generate accurate classification by utilising the scanning pixel camera's ability to focus only on these areas.

2.3 Visual Explanations

The field of explainable AI has developed several techniques to see which areas of the input are important for predicting the output. These methods are designed to give visual explanations, making it easier to understand how the network came to such a conclusion. An example of these explanations are shown in fig. 2.1. These include guided backprop [55], which looks at the products of the activations and their gradients as they flow back through the image. GradCAM, a more recent approach [52] improves on these visualisations by making them class discriminative. Here the gradient of a class score at each feature layer is produced by back-propagating the gradient of one for the target class and zero elsewhere. These are then summed to produce an activation map for each class.

AblationCAM [48] improves on these visualisations while also removing the requirement to calculate gradients. This is done by scaling the activations at each layer by a factor generated from the resulting class score and the score if those activations had been zero. These activations are then combined in the same way as GradCAM to produce better localised results. These visual explanations tell us that useful information is concentrated in certain areas of the image. It

is natural to ask whether we can discard the unimportant regions rather than simply downweight their importance.

These techniques demonstrate that a network will naturally learn to prioritise the information found in certain areas of the input, similar to how mammalian vision is attracted to areas of change and motion. This behaviour can be explicitly encouraged in order to improve performance of a variety of models through the use of attention mechanisms.

Convolutional neural networks (CNNs) can also be trained to recognise which areas are important as a separate task known as saliency estimation. Here a network produces a saliency heatmap similar to the visual explanations above. To gather ground truth data for this, datasets such as DUTS [64] and CAT200 [34] gather eye-tracking data from human participants. These are then processed to form heatmaps showing where people would look at a given image, which the networks are then task with reproducing.

The CNNs that are designed for this task generally follow a U-Net [49] architecture. Some such as Pyramid Feature Attention [75], uses a context-aware pyramid feature extraction module to capture the rich context features to allow better localisation of the heatmap. Others [73], take a more physics based approach, based on the rules of eye motion, in order to determine which areas of an image would be salient.

Whatever the chosen method, saliency estimation provides a method of estimating how important a pixel is to describing the contents of the scene. This may be of utility to this thesis, as the importance of a potential sample needs to be established in order to decide whether to sample it.

2.4 Attention Mechanisms

Neural networks can be given an explicit mechanism to prioritise different areas of an input. These are referred to as attention mechanisms and have traditionally been used to improve the effectiveness of sequence based deep learning problems. They are particularly of use in this setting as sequential data does not have the same spatial relationships that can be found in images.

Perhaps the most famous of these is the multi-head attention found in Transformer networks [59]. These have achieved state-of-the art results when applied to machine translation tasks. Multihead

attention allows the transformer to compare how self-similar a sequence is (self-attention) as well as how similar it is to an output sequences (cross-attention), which allows the model to be very expressive. However, the use of an attention mechanism for sequence based tasks is not a new idea, having been previously used to improve results with an RNNs [2]. Similar techniques have also used attention on sequences directly without the use of any RNN based processing [46].

Multihead attention has been successfully applied to the image domain [16] to uncover complex relationships between different image based features. This also manages to achieve state-of-the-art results on image classification tasks. This is likely due to the fact that although there is strong spatial coherence within an images, there is still information that can be gained by analysing the relationship between disparate areas of an image

These methods rely on soft attention, which only re-weights how valuable different features are and is unable to completely remove their influence. Yue *et al*. [72] demonstrates that it is possible to gain further accuracy by actively ignoring areas of an input. This is done by iteratively shifting patches toward the relevant areas of the image to provide better tokenised information to the vision transformer. This shifts the focus of the classifier away from redundant information, and increases the accuracy of a classifier by 3.8% while reducing the number of parameters required for the task. This still has the issue of having to view the entire input in order to decide where to shift the attention to so is not able to operate in a predictive manner.

Attention mechanisms can be applied at any layer in a network and tend to yield greater performance when done so. This, along with the techniques described in section 2.3, demonstrate that some incoming pixels (and their associated activations) will be more valuable in producing an accurate output. In contrast to the above input agnostic attention mechanisms, this thesis will only focus on the incoming pixels of the image to determine which should be utilised by the network.

The closest works to those proposed in this thesis are hard visual attention mechanisms such as SACCADER [19]. These explicitly avoid looking at areas of the input when performing a computer vision task. This divides the input into a series of glimpses that are independently processed before finally being combined in order to produce the final output for the whole image. Our work differs as we estimate the usefulness of each pixel to a task network and then forward

only the important pixels to the task network, rather than operating on a set of features that are produced by a series of glimpses. Furthermore, SACCADER is bound by simplification 1 of this thesis, and has to look at the entire input to decide where to place the glimpses. Our work goes beyond this, and is able to predict where to place samples without observing what is at that position.

2.5 Sampling techniques

In almost all domains, when the number of samples in a signal must be reduced, regular downsampling is the most commonly used method as it is easy and efficient to implement. However, this limits the maximum observable frequency [53] to be uniform across the image, and thus reduces the information available for downstream tasks. Regular downsampling introduces aliasing of the signal, however this can be eliminated using stochastic sampling [10], where samples are taken at random intervals according to a distribution.

With a greater understanding of the nature of the data being sampled, it is possible to further reduce the samplerate without sacrificing as much information content. This can be demonstrated with Level-Crossed sampling [50]. Here a sample is only taken when the change in input signal has crossed a particular threshold. It is then possible to reconstruct the original signal using polynomial interpolation.

Techniques to sample pixels based on their content rather than position for the purposes of image reconstruction has also been developed. An example of this includes Mixed Adaptive-Random sampling [69]. This combines a random sampling pattern with edge information in order to sample an image, which can then be later reconstructed. This was able to achieve 0.9 Structural Similarity (SSIM) when reconstructing.

Sparse sampling techniques are used in domains such as scanning microscopy, where imaging may take a long time to perform. An example of this is SLADS-Net [74], which takes the set of samples and their positions and gives the positions of where the next set of samples should be taken, deciding based on which samples would be most likely to reduce the distortion in the image. This process is repeated until the desired number of samples is taken or the desired

output quality has been reached. This was able to decrease the image acquisition from hours to minutes.

These methods are generally designed for reconstruction of the original signal. This thesis will look at sampling for the purposes of a downstream computer vision task such as image classification. In this case it may gathering enough samples for an accurate reconstruction may not be necessary, as only the salient regions of the image are needed in order to form a decision. In fact, removing the requirement for accurate reconstruction may prove useful as distracting information in the scene is no longer required to be imaged so is not able to disrupt the decision making process.

In a similar vein, Kaiming He *et al*. [24] discuss masking the inputs to auto-encoders before regenerating the original as an output as a method of self-supervised pre-training. While the work in this thesis is related in the sense that the neural networks designed here will have to develop some understanding of an input based on incomplete information, the difference is again the removal of the reconstruction step.

Chapter 3

Establishing Useful Samples

The end goal of this thesis is to provide a method of predicting useful samples that a scanning pixel camera can take for a downstream computer vision task such as image classification or semantic segmentation. Ideally, the number of samples taken should be minimised to do this in order to perform the imaging in the fastest possible time. This means that any method must be able to deduce if a sample will be good based only on information it has on other samples. As an initial problem, this is difficult to solve in a single step as we first have to know what makes a good sample and then how to predict where to take it from. Therefore, to simplify this process, this chapter will deal with establishing which samples are useful to take, and will tackle the prediction step in future chapters.

To do this we will produce sample map ($\mathbf{S}$), given prior knowledge of the full image ($\mathbf{I}$). This sample map will be used to create a scanning pixel camera image ($\hat{\mathbf{I}}$) following the method presented in section 1.3. The scanning pixel camera image, as simulated according to eq. (1.10), using the cross-correlation operator $*$ with an overlap kernel ($\mathbf{K}$). The resulting scanning pixel camera image will then be fed into a downstream computer vision task ($\mathcal{T}$) and the result will be evaluated using the metric for a downstream computer vision task ($\mathcal{M}$).

$$\mathbf{S} = \underset{\mathbf{S}}{\operatorname{argmax}} \ \mathcal{M}\left(\mathcal{T}\left(\mathbf{K} * \left(\mathbf{I} \odot \mathbf{S}\right)\right)\right) \tag{3.1}$$

Ideally, eq. (3.1) should be achieved using the lowest possible number of samples taken from a scene (n). However, there will be a minimum amount of samples that will have to be taken

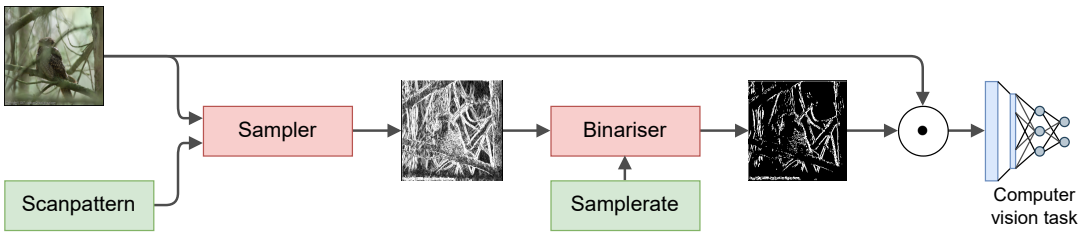


Figure 3.1: An overall architecture diagram of the system. The sampler produces probabilities that work at all sample rates and the user selects the one desired for the downstream task. This generates a sample map that is used to mask the input image using the hadamard product $\odot$

in order for the task network to achieve reasonable performance. Given that the data that the task networks are being trained from comprises of images of different resolutions, containing a different number of maximum number of samples that can be taken from a scene, the minimum required samples is defined as a target sample rate (κ_a),

$$\kappa_a = \frac{n}{N} \tag{3.2}$$

It then follows that $\mathbf{S}$ should obey the following constraint.

$$\frac{1}{N}\sum \mathbf{S} = \kappa_a \tag{3.3}$$

During this chapter we will attempt to establish a good value for κ_a. This may not be a fixed value, as $\mathcal{M}$ is not likely to scale linearly with different target sample rates. Furthermore, what an acceptable lower bound for accuracy is context dependant.

3.1 Methodology

The architecture of the proposed system is shown in fig. 3.1. This operates in two stages, a heatmap generator ($\mathcal{S}$) and a heatmap binariser ($\mathcal{B}$). The heatmap generator is given access to all samples from image ($\mathbf{I}$) and produces a sample heatmap ($\mathbf{H}$). This will be the same size as $\mathbf{I}$ and each element will represent the probability that that position should be sampled. The heatmap binariser will then take $\mathbf{H}$ along with a target sample rate (κ_a) and use it to produce sample map ($\mathbf{S}$). The sample map will then be used to produce a scanning pixel camera image that can be passed into a downstream computer vision task. This can be expressed as,

$$\mathbf{S} = \mathcal{B}\left(\mathcal{S}\left(\mathbf{I}\right), \kappa_a\right) \tag{3.4}$$

As we do not yet know what a good value for κ_a should be, any sample heatmap produced should be capable of generalising to any value of κ_a so that they can be used to establish this value. This means that the distribution of values in $\mathbf{H}$ should not be too highly skewed.

3.1.1 Heatmap generator Design

We propose two different heatmap generators that are capable of generating sample heatmaps. A simple, computationally efficient design known as SAUCE and a deep learning based design known as DeepSAUCE.

SAUCE

A simple method for sampling would be to focus on areas where a sample value (s) is changing. Areas with comparatively less change, such as the sky, are likely to contain less relevant information, and therefore should be represented with fewer pixels. Through the accumulation into a scanning pixel camera image, the sampling procedure preserves the spatial information of each sample. Therefore it should be possible for a downstream task to learn that a single sample is representative of the unsampled areas around it. We also take into account the distance in the sampling position (i), as moving a significant distance implies we are looking at something new and therefore we should take a sample.

These are combined to create a sample distance ($\mathbf{d}$) metric as follows,

$$\mathbf{d}\left(i\right) = \alpha\dot{\theta}\left(i\right) + \beta\Delta\mathrm{s} + \gamma \tag{3.5}$$

where $\dot{\theta}\left(i\right)$ is the change in scan position, Δs is the change in intensity since the previous sample. α, β and γ are learnable parameters, trained using the method discussed in section 3.1.3.

From this distance metric it is possible to calculate the values of sample heatmap as a probability using the variance of $\mathbf{d}$ (σ^2),

$$\mathbf{H}^i = 1 - e^{\frac{\mathbf{d}^i}{\sigma^2}} \tag{3.6}$$

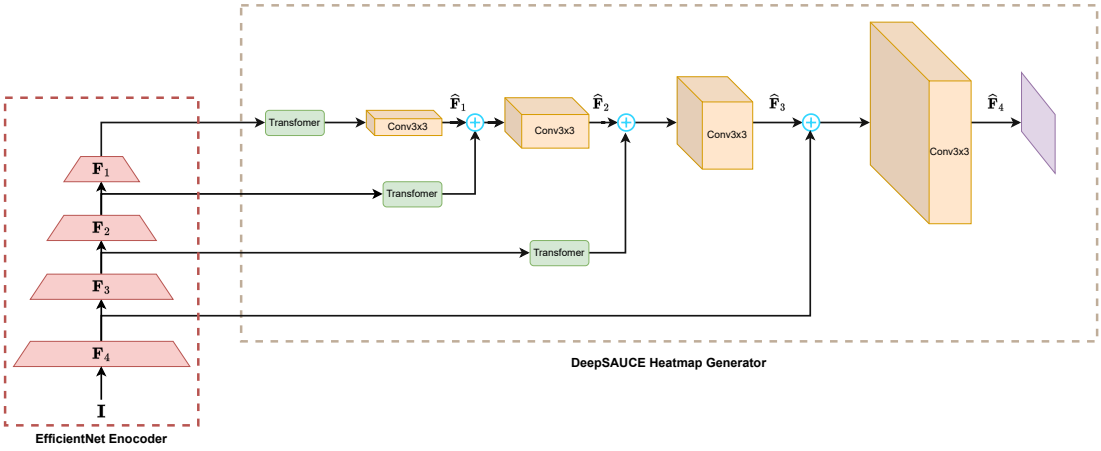


Figure 3.2: The DeepSAUCE architecture. Each intermediate feature representation ($\mathbf{F}$) is processed by a different stage of the heatmap generator. The branch for processing $\mathbf{F}_4$ does not contain a transformer layer due to computational constraints. Full details of the dimensions of the weights used can be found in appendix A.

DeepSAUCE

The second sampling scheme proposed is based on a deep learning model, as these have seen success in many other fields. This utilises an U-Net [49] inspired architecture with transformer modules on the skip connections to the decoder, as shown in fig. 3.2. As an encoder EfficientNet-b0 [57] is used, although any convolutional neural network (CNN) should be sufficient.

The image is passed into a series of feature encoder modules ($\mathcal{E}_{1..N}$) found in EfficientNet. Each feature encoder module takes the output from the previous encoder step to produce an intermediate feature representation ($\mathbf{F}$).

$$\mathbf{F}_i = \mathcal{E}_i(\mathbf{F}_{i-1}) \tag{3.7}$$

This is followed by a series of feature decoder modules ($\mathcal{D}_{N..1}$). These take the output of the previous feature decoder module and the output of the corresponding feature encoder module stage to produce heatmap features ($\hat{\mathbf{F}}$).

$$\hat{\mathbf{F}}_i = \mathcal{D}_i(\mathbf{F}_i, \hat{\mathbf{F}}_{i-1}) \tag{3.8}$$

For all layers excluding the last utilise transformer encoder layers [60] consisting of feed-forward networks and multiheaded attention. The transformer treats each 2D $\hat{\mathbf{F}}_i$ as a 1D sequence of

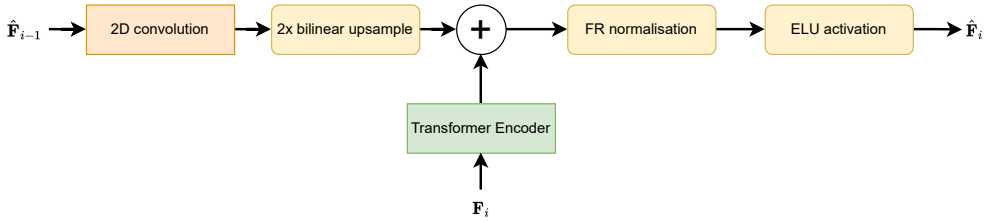


Figure 3.3: Each of the convolutional blocks produces the modified features ($\hat{\mathbf{F}}_i$) from the previous block's output ($\hat{\mathbf{F}}_{i-1}$) and the current set of features from the feature pyramid ($\mathbf{F}_i$). The convolution is followed by a $2\times$ upscale in order to maintain the correct size for the hadamard product.

pixels to produce an output. The 2D shape is then restored immediately after the transformer has been used. This allows a richer representation of more complex spatial relationships that could not be done using only convolution layers. In the final decoder layer the transformer encoder are not used as the size of the sequence makes it computationally infeasible.

After applying the convolution a $2\times$ bilinear upsample is performed to match the size for summing $\mathbf{F}_i$ to the result. We then apply a filter response normalisation [54], as this is a slightly more performant version of a batch normalisation, especially if there is need to train at low batch sizes.

To help maintain discrimination between the different parts of the sample heatmap, we then apply exponentially decaying ELU layers [9] rather than the more common clipping based RELU layers. This modifies the network so that its output block asymptotically approaches -1, rather than being clipped to 0. This effect is also achieved in the SILU layers [26] in the feature encoder module. The output of the ELU activations are then passed into the next convolutional layer as shown in fig. 3.3.

Once all the feature decoder modules have been processed a final convolution is applied in order to produce a single channel sample heatmap, representing the probability of taking a sample at that position. This can then be passed into the binariser to produce the sample map ($\mathbf{S}$).

3.1.2 Heatmap binariser Design

The heatmap binariser's task is to take the sample heatmap and produce a sample map the obeys the constraint in eq. (3.3). As $\mathbf{H}$ is a probability map (or in the case of DeepSAUCE can be converted into one using a sigmoid function), it is possible to use maximum likelihood estimation in order to convert $\mathbf{H}$ into $\mathbf{S}$ while attempting to maintain the value of κ_a. This produces the sample map ($\mathbf{S}$) as follows,

$$\mathbf{S}^i = \begin{cases} 0 & \mathbf{H}^i < 1 - \kappa_a \\ 1 & \mathbf{H}^i \geq 1 - \kappa_a \end{cases} \tag{3.9}$$

It is desirable that this process is differentiable as then it will be possible to train the heatmap generator based on the gradient from the task, thus making it select task specific samples. Therefore, to make the process in eq. (3.9) differentiable a straight-through trick is employed, as described in algorithm 1. This preserves the binary nature of $\mathbf{S}$ while making the gradient flow through $\mathbf{H}$ during the backward pass.

Algorithm 1 Straight-through trick

procedure STRAIGHT-THROUGH($\mathbf{H}$, $\mathbf{S}$)

$\mathbf{S} \leftarrow \mathbf{S} - \mathbf{H}$

Detach $\mathbf{S}$ from the computation graph

$\mathbf{S} \leftarrow \mathbf{S} + \mathbf{H}$

Return $\mathbf{S}$

end procedure

Sample Heatmap Normalisation

Equation (3.9) is not guaranteed produce a sample map that meets the target sample rate. In cases where the heatmap generator is learnt, it is possible to encourage it to do so by using a loss as described in section 3.1.3 and sample rate scheduling described in section 3.1.4. However, this may not necessarily be enough, and an internal mechanism is desirable.

Figure 3.4a shows that if no constraints are applied to the sample heatmap it becomes heavily binarised, making it only suited to a single κ_a. This is due to a heavily skewed distribution of

sample probabilities, as it is possible to achieve the target sample rate as long as n samples have a probability greater than κ_a. Thus a network producing the sample heatmap can produce a binarised mask as it has no reason to fill the full range of sample probabilities. It is possible to produce a model for each value of κ_a, however this is time consuming so an alternative solution is needed.

The binarisation of the sample heatmaps can be partially mitigated by training at multiple sample rates. However, it is difficult to define when enough examples have been seen to ensure generalisation. Equally, a loss would be difficult to define as the sample heatmaps need to maintain some form of skewness in order to be discriminatory for the downstream computer vision task.

The solution used in this thesis is to have the heatmap binariser rescale $\mathbf{H}$ based on the value of κ_a. To do this, the logit of κ_a is added to $\mathbf{H}$,

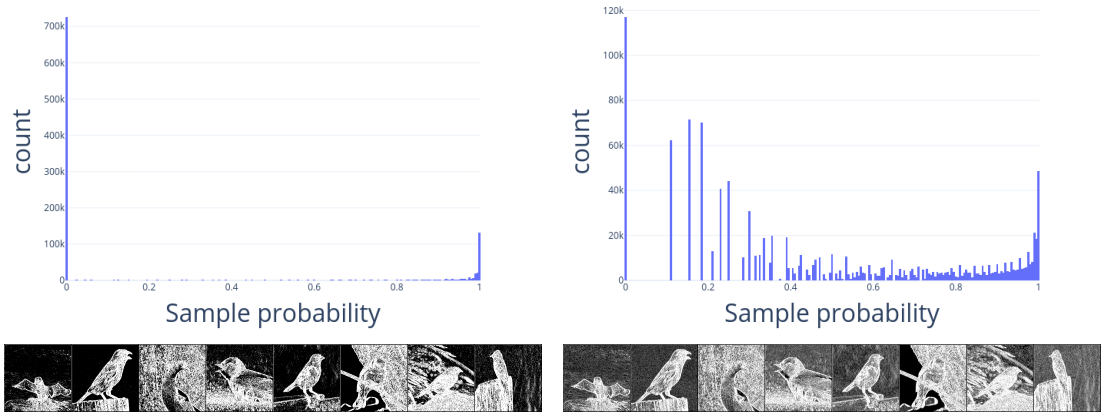
$$\mathbf{H} = \mathbf{H} + \ln\left(\frac{n}{N-n}\right) \tag{3.10}$$

This modifies the amount of samples taken by ensuring that each sample will have a base probability corresponding to κ_a. $\mathbf{H}$ will then increase these probabilities in important regions, and decrease then in unimportant ones. Note that this process has no effect on the gradient that flows through $\mathbf{H}$.

This is then passed through a sigmoid and normalised so that it will fall into a $[0, 1]$ range, with a guaranteed zero value, ensuring that it can generalise to even small values. This results in a wider range of values for $\mathbf{H}$, as shown in fig. 3.4b, which gives better performance over a wider range of sample rates.

3.1.3 Training scheme

The sampler is simultaneously trained along with the downstream task in an end-to-end fashion. The training of the downstream task is not modified other than changing the input from $\mathbf{I}$ to $\hat{\mathbf{I}}$. So that the downstream task is able to operate on sampled images, we allow all the layers of the task network to train. This allows for relevant intermediate feature representations to be produced from a sampled input.



(a) Without normalisation

(b) With normalisation

Figure 3.4: Distribution of sampling probabilities with and without the normalisation in eq. (3.10). The normalisation produces a wider range of probabilities which helps with generalisation to other target samplerates at runtime.

To apply the constraint stated in eq. (3.3), an L1-loss is applied to the sample map,

$$\mathcal{L}_\kappa = \left\| \kappa_a - \frac{1}{N} \sum \mathbf{S} \right\| \tag{3.11}$$

As the system is end-to-end differentiable, the loss being applied to the task network ($\mathcal{L}_{\text{task}}$) will also train the sampler, allowing it to find samples specifically useful for solving the task.

3.1.4 Sample rate scheduling

At inference time κ_a is supplied by the user. This means the system is flexible and sample rates can be adapted on the fly based on application requirements. To support this, during training the target sample rate is selected according to a scheduling function ($\mathcal{D}$).

The simplest case where a set sample rate for inference is known a priori we can simply specify,

$$\mathcal{D}_{\text{const}} = \kappa_a \tag{3.12}$$

This will fully optimise performance at the defined target rate but is likely to negatively impact run-time generalization if the user desires a sample rate that is not κ_a.

Alternatively if the only target is to minimise the sample rate as much as possible, we can define,

$$\mathcal{D}_{\exp}(q) = \gamma^q \tag{3.13}$$

where q is the number of epochs and γ is the decay factor. This allows the system to begin training in the easier domain where more information is available before slowly learning to focus its computation efficiently, while simultaneously ensuring that the system is able to adapt to different sample rates.

Alternatively, we can randomly sample the target sample rate,

$$\mathcal{D}_{\mathrm{rng}} \sim [0, 1) \tag{3.14}$$

so that the heatmap generator is never able to overfit to one sample rate. This ensures that it must retain the ability to produce sample heatmaps that generalise to all sample rates.

We can also sample form other distributions, such as the beta distribution ($\mathcal{D}_{\mathrm{beta}}$) in order to bias toward a specific sample rate, while maintaining some generality. This may also be achieved by combining D_{rng} and $D_{\exp}$,

$$\mathcal{D}_{\mathrm{exprng}}(q) \sim [0, \gamma^q) \tag{3.15}$$

3.2 Image Classification Application

The first task the samplers are tested on is image classification. Here the task is to predict the correct label $\mathbf{y}_i$ for each input image $\mathbf{x}_i$. As a metric the average accuracy for m inputs is used,

$$\mathcal{M} = \frac{1}{m} \sum_{i=0}^{m} 1\left(\hat{\mathbf{y}}_\mathbf{i} = \mathbf{y}_\mathbf{i}\right) \tag{3.16}$$

where $\hat{\mathbf{y}}_\mathbf{i}$ are the predicted label. In a deep learning context, models for this task are normally trained with a cross-entropy loss.

Image classification is one of the first tasks where a convolutional neural network (CNN) was shown to be highly effective, with AlexNet [32] scoring significantly higher than other methods at the time for classification of the ImageNet dataset [13]. The strong spatial coherence of the convolutions combined with the information compressing effects of the pooling layers used makes it an ideal method to tackle a classification task.

Since then a large amount of improvements have been made to CNNs, though the basic idea remains the same. These include the use of batch normalisation to increase generalisation [29], residual connections [25] to allow the construction of deeper and more powerful networks, and optimisations done to architecture and hyperparameters of the network [57]. Today CNNs are a mature technology with the latest transformer enabled networks [17] able to achieve over 90% classification accuracy.

3.2.1 Training & Evaluation protocols

As a classifier network, EfficientNet-b0 [57] that has bee pre-trained on ImageNet [13] is used. The following hyperparameters were experimentally determined. All models are trained using the ADAM optimiser [31] with a learning rate of 0.001, with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. All models were trained for 100 epochs. An exponential decay of 0.98 was applied to the learning rate. During training, each example image had its target sample rate (κ_a) selected according to eq. (3.14). Full details associated training parameters can be found in chapter B.

Evaluation occurs across a range of sample rates, and plot the resulting for the metric for a downstream computer vision task for each sample rate. Ideally, high task performance should be achieved across all sampling rates, especially at the more challenging low sampling rates.

We compare our results to other sampling methods, as well as a fully-sampled input to gauge the ability of the task model as-is. These include simple methods such as uniform downsampling and random sampling. More complex predictive methods such as level-crossing sampling [50] and mixed adaptive-random sampling (MAR) [69] are also used. These will be referred to as the baseline samplers.

Initially, classification is performed on the Caltech-UCSD Birds 200 dataset [62], which comprises of images of single birds. While this is a relatively small dataset, it provides a good way to evaluate the efficacy of our technique. The images are such that the bird that needs to be classified has an obvious silhouette in the centre of the frame. However, the task is not as easy as the similarities between birds of different species substantially increases the difficulty of the task.

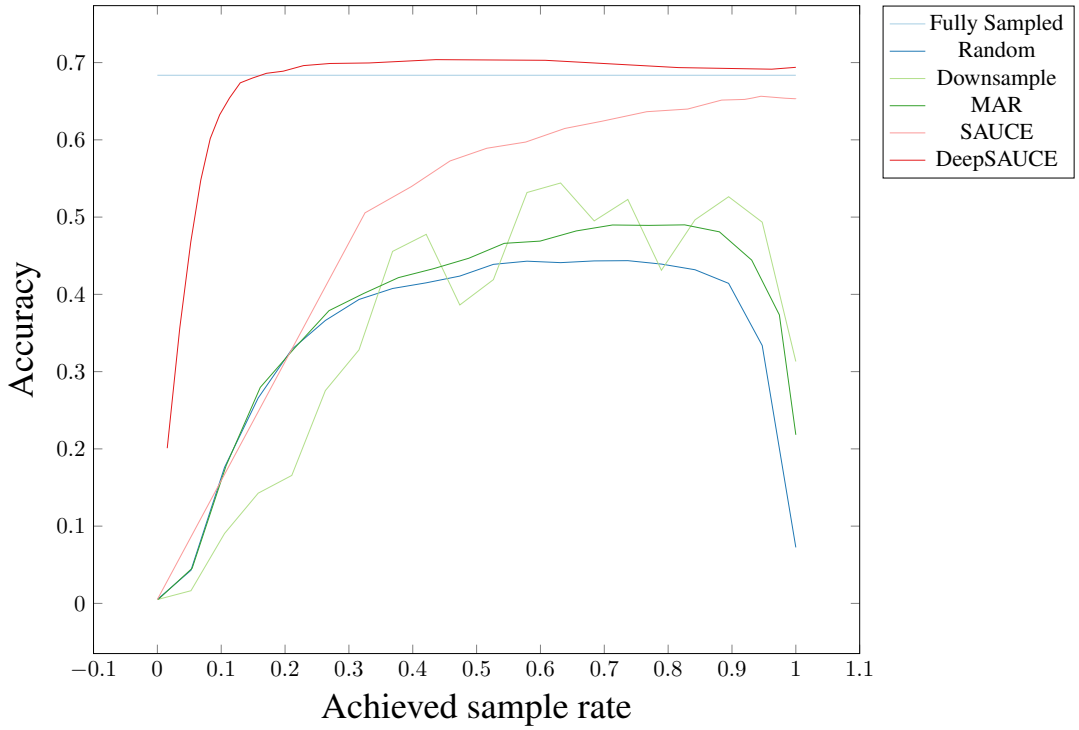


Figure 3.5: Results of our samplers on Classification of the Caltech-UCSD Birds 200 dataset

3.2.2 Results

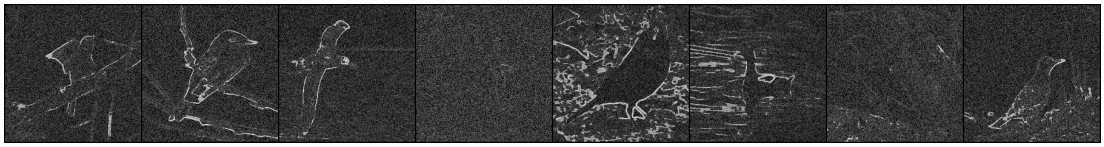
From fig. 3.5 both SAUCE and DeepSAUCE outperform the baseline samplers, with DeepSAUCE only having a 18% drop in accuracy while at a 20% sample rate. This shows a significant amount of samples can be ignored while still completing the classification task, which is useful for a scanning pixel camera. It is however unclear what a good value for κ_a should be when performing this task in the real world. If we wish to preserve the accuracy of the model then κ_a should be 0.61, however lower values are possible if the corresponding drop in accuracy is acceptable.

Our methods also outperform the more complex baselines of MAR and level-cross. This is likely because these will select samples that are designed for reconstruction, which are not necessarily the same samples that will be useful for the classification task.

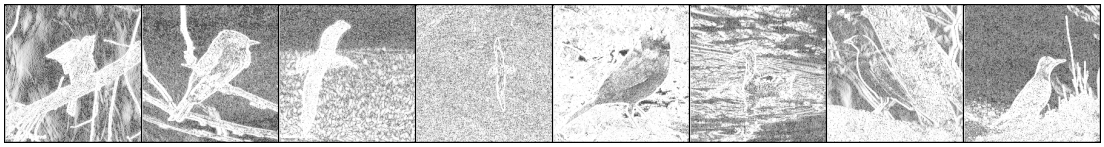
Looking at the qualitative results in fig. 3.6, it is notable that both SAUCE and DeepSAUCE capture significantly more detail in their sample heatmaps, even though they all highlight the location of the bird to some degree. This allows for a wider definition of what a useful sample

(a) Input image

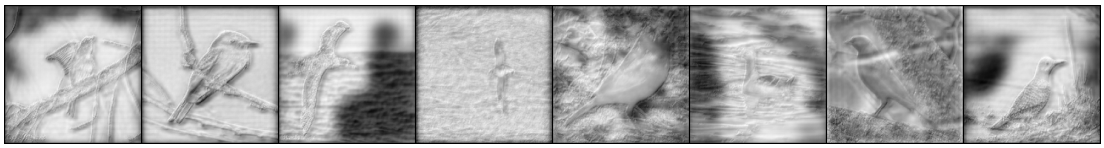
(b) Level crossing



(c) Mixed adaptive random



(d) SAUCE



(e) DeepSAUCE

Figure 3.6: Example sample heatmaps from the different heatmap generators. These examples are used for classification, other tasks may have different sample positions.

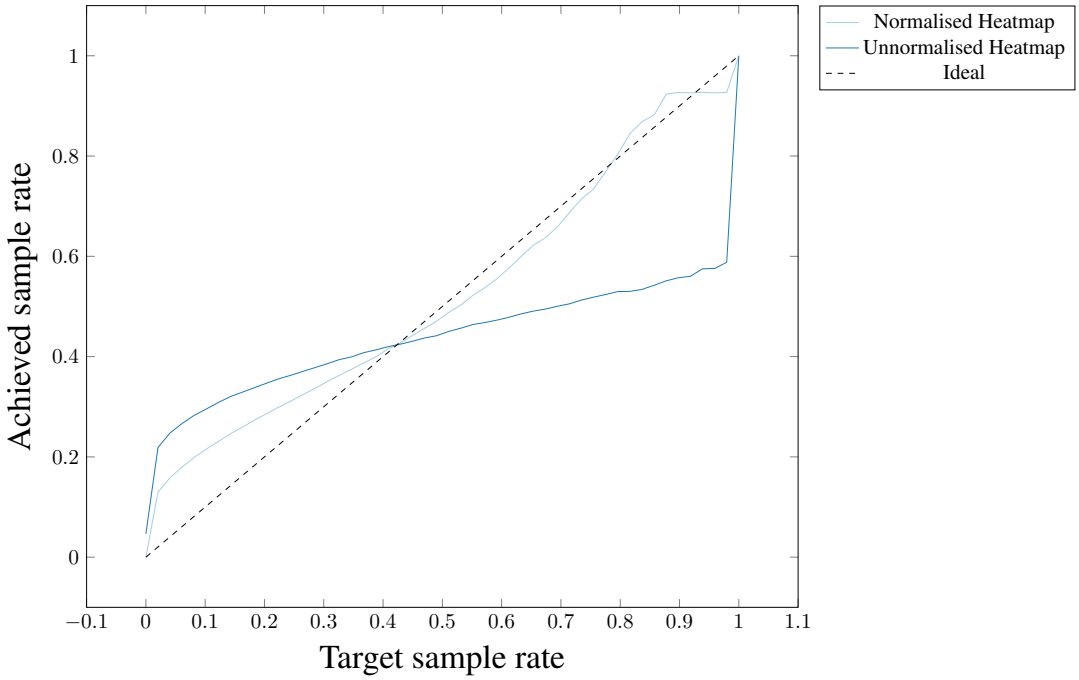


Figure 3.7: The effect of normalisation on SAUCE

is, which can be exploited by the task network. Notably, DeepSAUCE appears to produce a two stage sample heatmap, showing the transformers ability to capture both global and local information.

Impact of normalisation

Figure 3.7 shows the effect of the normalisation in eq. (3.10). It is clear that when it is applied the achieved samplerate more closely matches the desired κ_a, especially at the extremes. This means that the normalisation is key in allowing the system to generalise to a range of sample rates even though it has been exposed to these through the samplerate scheduling during training.

Comparison of samplerate schedulers

Although the sample heatmaps of SAUCE are designed to be general, training at different target sample rates allows for more generalisation. In experiments so far this was achieved by randomly selecting the samplerate per iteration. Figure 3.8 shows the effects of using the different samplerate schedulers described in section 3.1.4.

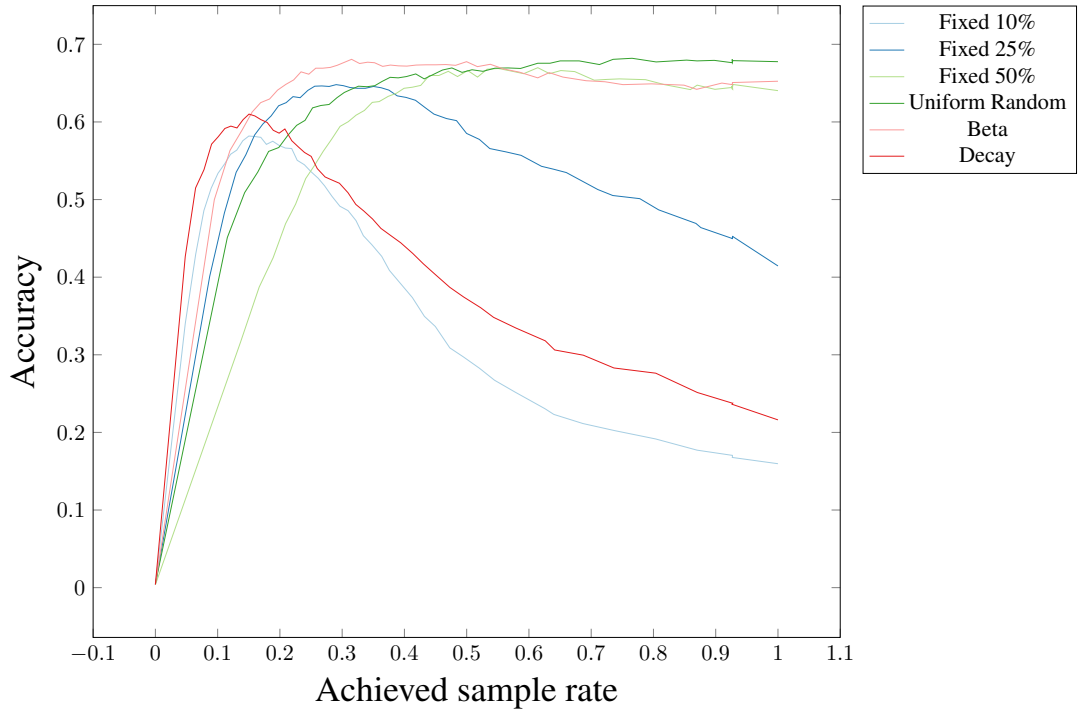


Figure 3.8: Results for Classification of the Caltech-UCSD Birds 200 dataset using different samplerate schedulers

The fixed rate samplers perform best at their target samplerate but unsurprisingly do not generalise as well as the others when asked to produce sample rates that were not seen during training. Decay does not generalise as well to lower sample rates. This is likely due to the classifier being biased by early epochs, so it tends towards a solution that prefers having access to more samples. The beta distribution behaves similarly to the uniformly sampled samplerate, however it is able to perform slightly better at lower sample rates as it has been more frequently exposed to this. This comes at the cost of performance at higher sampling rates, which it will have been exposed to less of. Based on these results, we will be using a random κ_a for all future experiments.

3.3 Using Saliency

As mentioned in chapter 1, humans are able to focus on the salient areas of an image. These saliency maps are equivalent in nature to our sample maps. Therefore, an alternative way of building a heatmap generator is to break down the process into two steps. Firstly, the heatmap

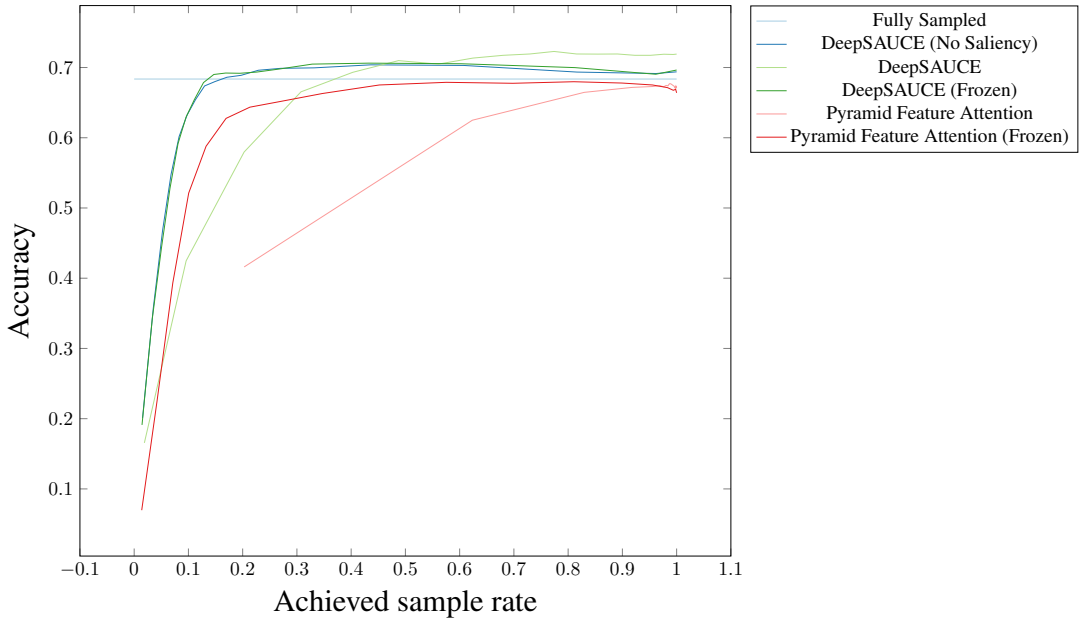


Figure 3.9: Performance of Saliency based models for classification on the CUB200 dataset

generator is trained to complete a saliency estimation task. Once this has been completed the heatmap binariser is attached to the end and trained for the downstream computer vision task.

To train such models saliency data from the DUTS [64] dataset is used. This dataset provides ground-truth saliency heatmaps based on human gaze, allowing the model to be trained as a regression. We borrow the combined regression and edge loss from [75] in order to regress accurate heatmaps. This performs a binary cross-entropy loss on both the resulting saliency map and the Laplacian of the saliency map to ensure that edges are preserved effectively. This also allows us to use any saliency detection model as a sampler, so as a comparison of the Pyramid Feature Attention Network [75] is used. DeepSAUCE is also reused for the prediction of saliency.

3.3.1 Classification Results

Figure 3.9 shows the accuracy when the saliency pretrained models are then trained to sampling for classification. For each saliency model two heatmap generators are produced, one where we freeze the saliency model after training and one where we allow it to be further influenced by the gradient from the task network. As the results show, there is a significant increase in the

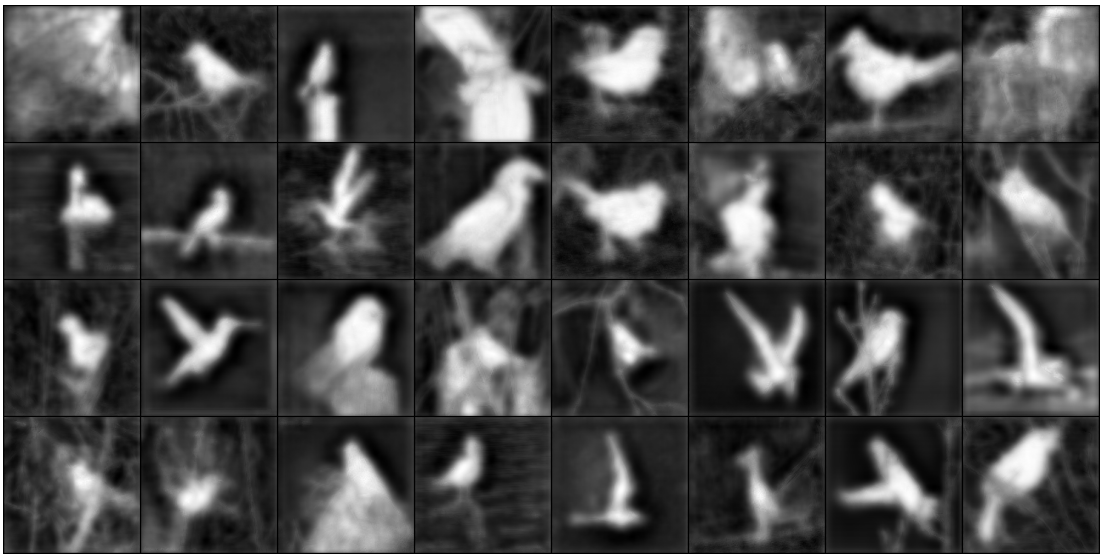


Figure 3.10: Sample heatmaps produced by DeepSAUCE when it has been pre-trained for a saliency estimation task.

performance of the sampling when using DeepSAUCE, being able to maintain a consistently high accuracy until it has less than 15% of the total possible samples. The nature of this result allows us to clearly state that a good value for target sample rate (κ_a) for performing this task in a real-world scenario is 0.15.

It is also worth noting that these high results are only achieved when the heatmap generator is frozen after training on the saliency data. This implies that it is easier for the downstream computer vision task to adapt to a salient image than it is for the heatmap generator to learn what is useful for the task. Therefore, going forward it is justifiable to use external sources of supervision in order to guide the training of the heatmap generator.

Looking at some qualitative results in fig. 3.10, we can see that they are a lot less focussed on detail and are more concerned about taking into account the overall area that may be useful. This may explain the increase in performance as it is difficult to determine if a single sample is useful, but samples taken from a salient area should always provide some relevant information, thus the heatmap generator does not need to pay attention to the fine detail and only to deciding which areas are relevant.

3.4 Expanding task coverage

In order to demonstrate that this technique is applicable to a wider variety of data and task settings, the models presented in this chapter are applied to additional classification datasets as well as additional tasks. Any additional training parameters used for these tasks can be found within appendix .

3.4.1 Classification datasets

As additional classification datasets we use the ImageNet [13] and iNaturalist [28] datasets. These provide much more challenging classification tasks, over a wider variety of classes. Both saliency trained models and the models trained in using only information backpropagated from the task network are used for this.

In fig. 3.11 and fig. 3.12 we compare our results to those of other samplers. While both demonstrate the superiority of the methods suggested in this chapter, they give differing conclusions. For the iNaturalist dataset the saliency based models are still clearly better, although they are unable to maintain this at sample rates below 0.4, giving a significantly higher value of target sample rate (κ_a) to maintain accuracy.

The ImageNet results tell a different story, here the DeepSAUCE model is able to achieve higher accuracies at lower sample rates but at higher values the saliency trained model performs better. This is likely due to the fact that ImageNet can apply labels that describe the scene as a whole, rather than objects in a scene. For example, an image labelled beach or dock may contain a boat. The saliency model that the sampler uses may consider the boat the most salient object, and thus focus the sampling on that, leading to an incorrect classification. At higher values of κ_a this will not matter as sufficient detail surrounding the boat will allow the classifier to pick the correct class. However, when samples are scarce, the fact that the non-saliency trained model puts more emphasis on surrounding areas allows there to be sufficient information to perform the classification correctly.

We also note that the saliency based models are unable to achieve the lowest κ_a values. Although effective, there is no guarantee that the normalisation in eq. (3.10) is able to achieve a given κ_a

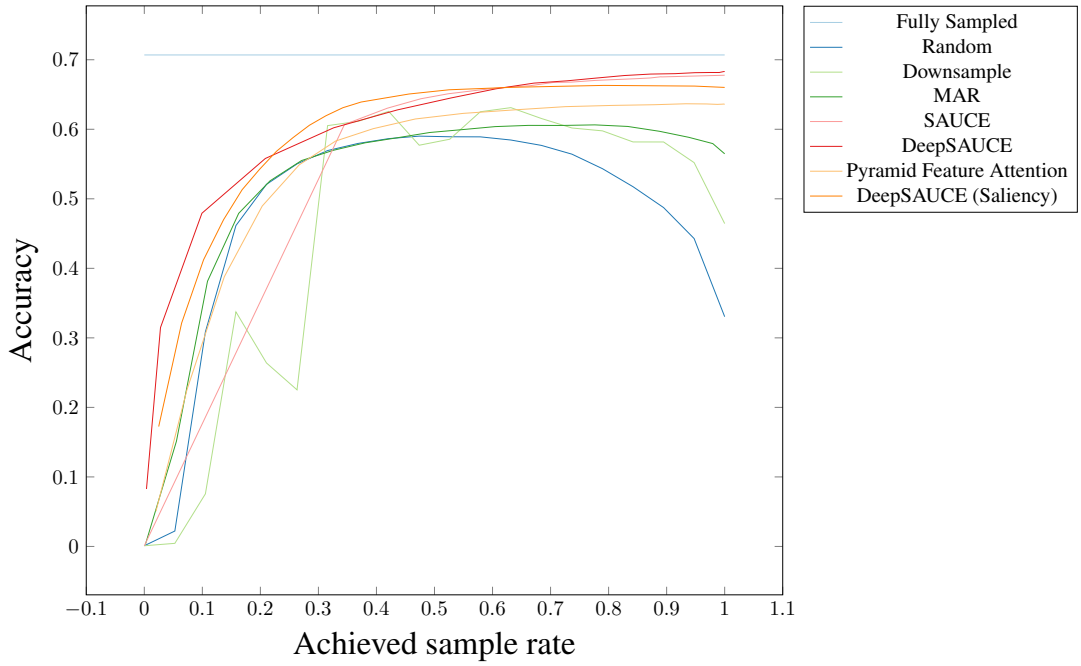


Figure 3.11: Results of our samplers on Classification of ImageNet dataset [13]

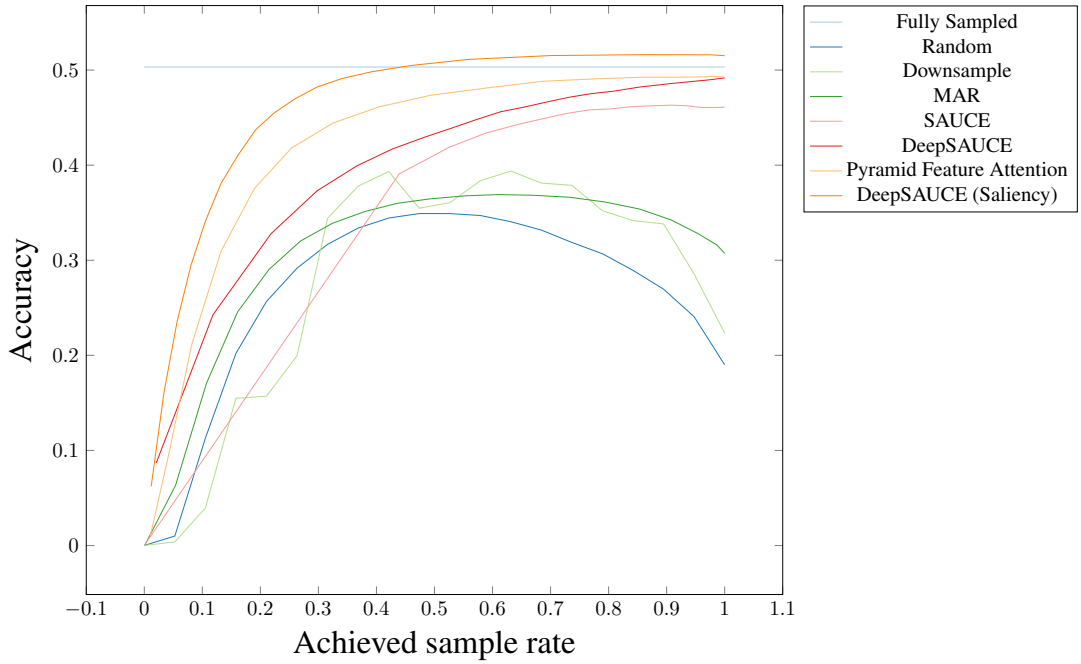


Figure 3.12: Results of our samplers on Classification of iNaturalist dataset [28]

and if the produced sample map contains high enough values then the lower sample rates will be unattainable.

Interestingly, some models are able to exceed the fully sampled baseline. At lower sampling rates this may be due to the heatmap generator ignoring distracting samples that may result in an incorrect classification. However, this result is maintained when κ_a is at 1.0. This is because the task network will be seeing partial images during training, which could be considered a form of the random erasing augmentation [76]. This has been shown to improve generalisation and overall accuracy when applied to many different computer vision tasks.

3.4.2 Instance Segmentation

In segmentation tasks, the location of each relevant object in the scene is indicated. This may be done by producing a bounding box or polygon around the object or providing a pixel-wise annotation as to the nature of that pixel. These may be done for the entire scene, which provides a label for all pixels and is know as semantic segmentation. Alternatively, only certain areas of the scene, which correspond to salient objects may require a label This is referred to as instance segmentation.

The label can also have an associated class, meaning that the challenge is to not only identify where and object is in a scene but also what that object is. In the case of pixel-wise labelling this involves assigning a label to each pixel. Additionally, instance segmentation requires the a distinction of different instances of the same class within an image.

Deep learning has been successfully applied to segmentation tasks with the advent of U-Net [49]. These work by using a convolutional neural network (CNN) to create a compact feature representation of the input, and then have a series of convolution based decoder steps that combine this information and scale the output in order to give a segmentation result. Since then many models [7, 8, 20] have improved on this design by making better use of the information found in the features provided.

In this section instance segmentation over is performed the MS-COCO [37] dataset. As a task this should be more difficult as the heatmap generator may need to focus on different areas of the scene, which may make it harder to maintain accuracy at lower values of κ_a. For this task

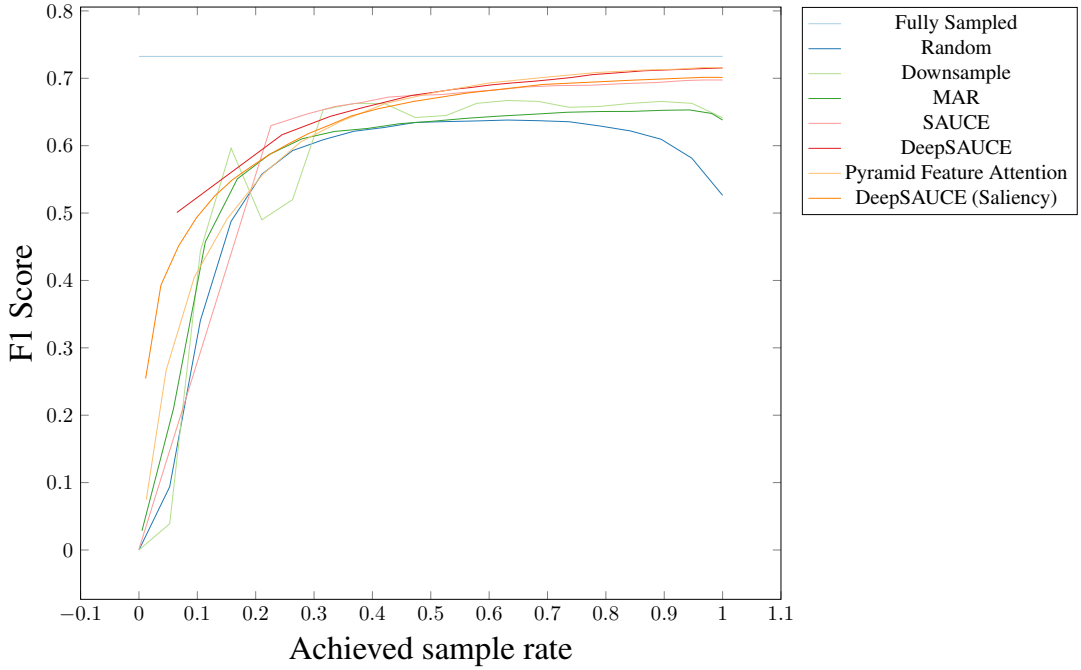


Figure 3.13: Results for instance segmentation of the COCO dataset

FPN [36] is used as a downstream computer vision task network. To evaluate, the F1 score is used as the task metric. The results for this are shown in fig. 3.13.

The results for this task again show that the saliency guided model does not perform as well when there are multiple objects in the scene that need to be focussed on. Though our model still outperforms the baselines at most target sample rates, the performance gap is significantly lower. This is likely due to the inherent difficulty of the task making it harder to optimise which samples should be chosen. Many models are good at preserving performance across a large range of κ_as, with SAUCE demonstrating the best retention of F1 score. This gives a surprising low value for κ_a for this task of 0.4.

3.5 Real-world system

A real world system does not have the luxury of imaging a scene twice. Although the aim this chapter is not to deal with this problem and to establish what a good sample would be, it is possible to modify the methods proposed in this to something that can approach a real world system. This can be done by producing the sampling for a high resolution scan based on an

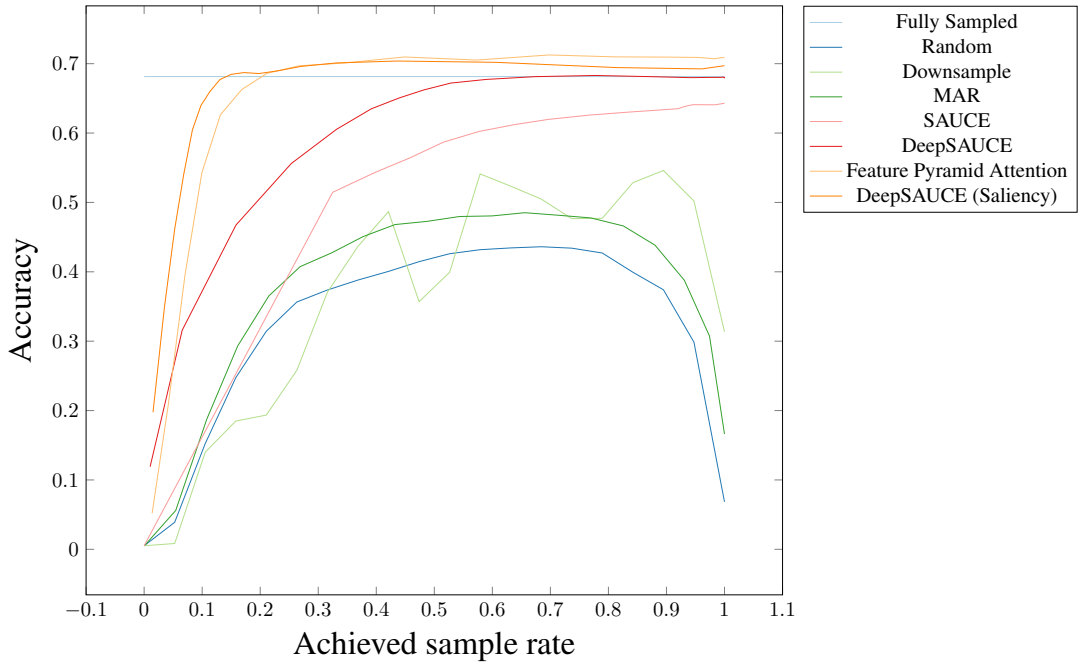


Figure 3.14: Accuracy when downsampling $4\times$ of the CUB200 dataset

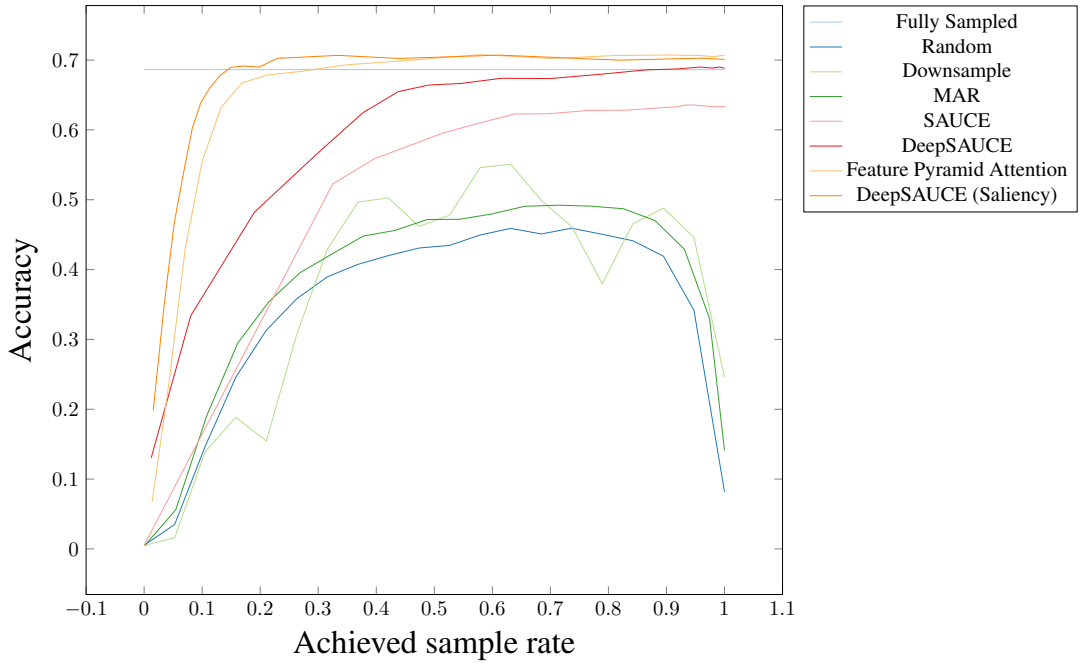


Figure 3.15: Accuracy when downsampling $10\times$ of the CUB200 dataset

initial low resolution scan. This would speed up sampling as the overhead from the first scan contains a lot fewer samples and therefore can be performed quickly.

To test that the models proposed in this chapter are still effective in this context, the sampler is trained to produce full sized sample maps from downsampled input images. The sample map is then applied to the full sized image in the manner of previous experiments in this chapter and obtain a result for the downstream computer vision task.

$4\times$ and $10\times$ down sampling are used to test this on our models on the CUB200 dataset and display the results in figs. 3.14 and 3.15. These show that the performance is maintained relatively well even though the high-resolution sample maps are being formed based on less information. There is a slight drop in performance although, which is more noticeable in the $10\times$ downsampling. This same effect can be seen in the other classification datasets, as shown in fig. 3.16. Additional datasets can be found within appendix C.

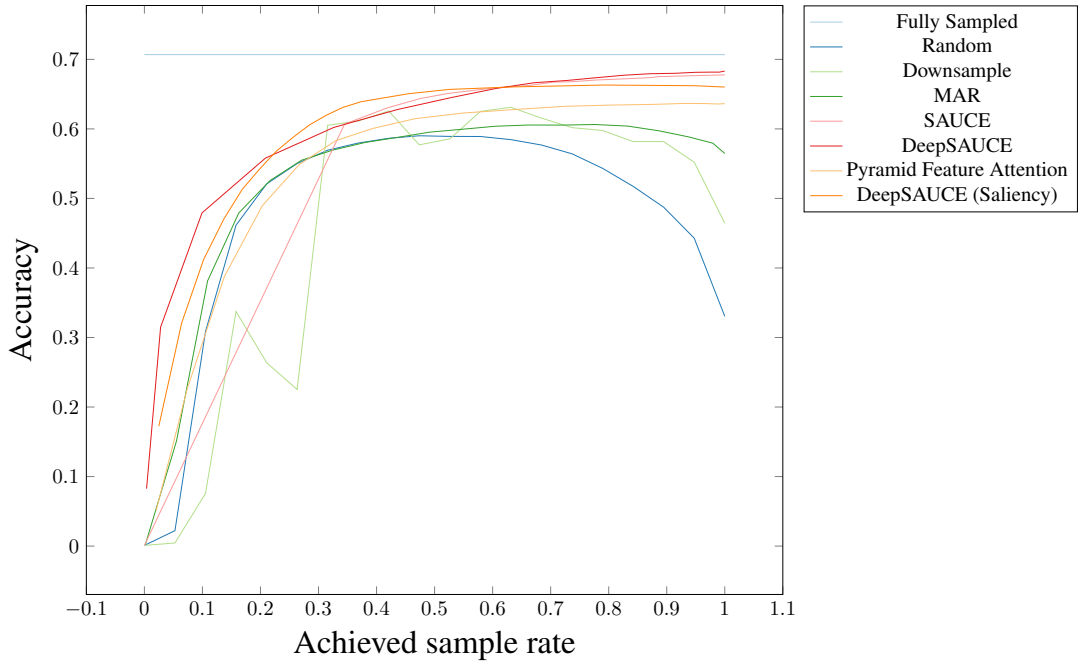
3.6 Conclusion

This chapter demonstrated that by training a sampling scheme specific to a downstream computer vision task, it is possible to significantly reduce the number of pixels viewed by that task while maintaining the efficacy of the task network. This can be done by either using information from the task or by pre-training based on saliency data. the latter being more effective when there is an easily identifiable target within the scene.

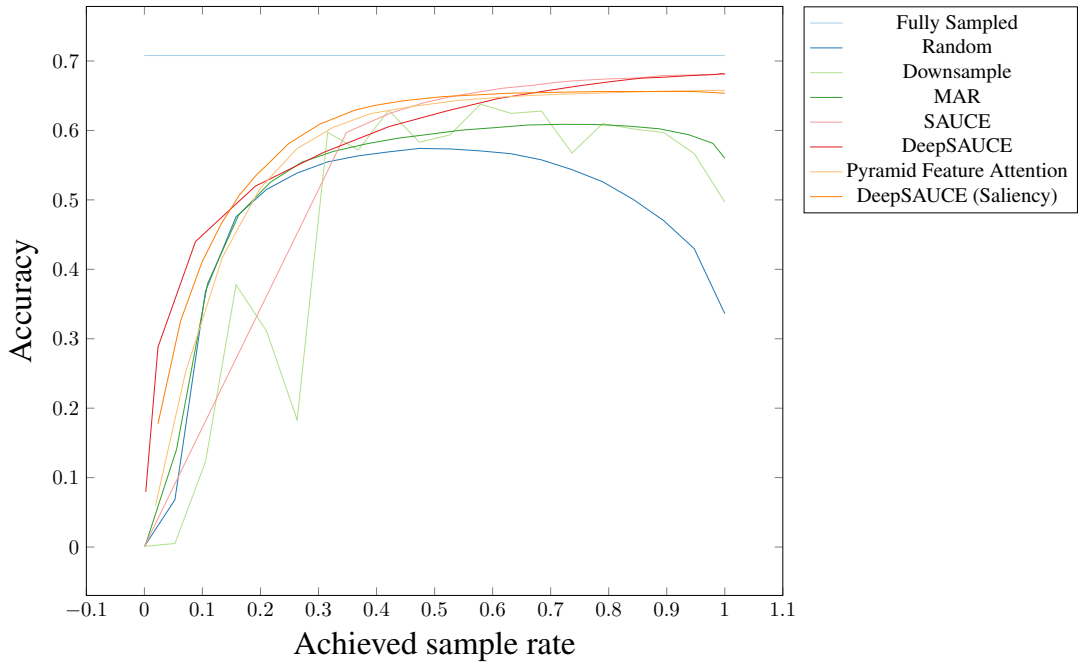
This significantly reduces the amount of data required to meaningfully represent the input for a downstream computer vision task. Such a reduction will be leveraged by future work in order to reduce the sampling time for a given image. This effectively demonstrates that a scanning pixel camera is useful in a computer vision context as it is possible to leverage its unique hardware capabilities.

Furthermore, it is possible to apply this to a more real-word scenario in a two stage process where a low resolution scan can be used to generate a high resolution sample map. This, retains the advantage of minimising the number of samples taken.

Overall the work proposed in this chapter allows for the generation of useful sample maps while obeying all the simplifications stated in the introduction. Future chapters will now build from



(a) ImageNet downsampling $4\times$



(b) ImageNet downsampling $10\times$

Figure 3.16: Accuracy when downsampling while classifying over other datasets

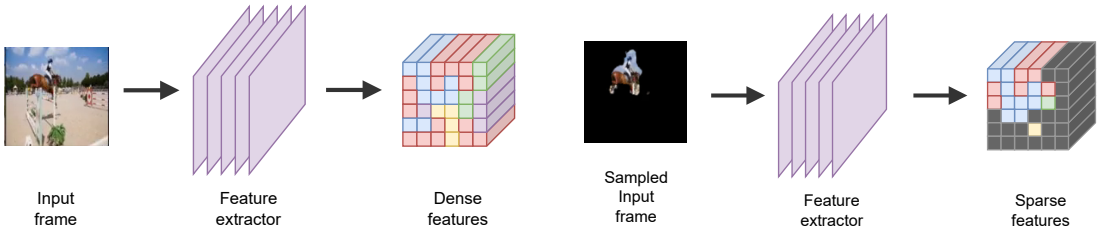
this, attempting to produce these sample maps in a predictive manner so that full information of the scene that is being imaged is no longer required.

Chapter 4

Where to look next: Predictive sampling for efficient computer vision

The previous chapter established a method of determining where a good sample is. However, to truly realise a scanning pixel camera, the sample positions need to be selected without prior knowledge of their contents. The previous chapter's approach to remedy this by having an initial scan was, though effective, inefficient. Even in the case where the initial scan was only 10%, this still comprises a significant overhead to acquisition time.

To remedy this, we now switch to dynamic scenes. Many real world problems, such as object tracking or pose estimation, require dynamic information. scanning pixel cameras are particularly



(a) Normal **(b)** Ours

Figure 4.1: When processing an input frame, a feature extractor will product features at all positions, which may include distractors (fig. 4.1a). Our method pre-emptively samples the input so that features will only be produced for the relevant areas of the image (fig. 4.1b)

suited to object tracking as the positions where the sensor is taking samples should follow the target object. In order to do this the heatmap generator will have to predict the locations of samples for the next frame based on the samples of the current frame.

This also alleviates the assumption that the scanning pixel camera is viewing a static scene as the scene will be changing between frames. To emulate this in simulation, scenes are taken from video, giving many parts of the framework developed in this chapter a temporal component (x_t). In doing so we move closer to realising a fully continuous sampling system, although it is still bound by taking a frame of samples at a time.

This chapter will focus on a general purpose algorithm to make the decision on where the samples in the next frames should be taken based on the previous (sampled) input. This allows for the best possible leverage of the transmission bandwidth and sensor capabilities, without complete prior knowledge of the scene. Additionally, the necessary training techniques for this algorithm will also be described.

In summary, this chapter presents the following contributions. Firstly, a hard attention mechanism to pre-emptively focus on important areas of the next frame and to avoid capture, transmission and processing of uninformative regions. This will be created using only information from frames that have already been seen. Secondly, a general framework that can augment any existing frame based video network with to apply this hard attention mechanism to increase efficiency.

4.1 Methodology

The proposed approach taken in this chapter consists of three main building blocks, as summarised in fig. 4.2. As the notation is now time dependant, subscripts will indicate the time step and superscripts will represent indices. At each frame, t, an image ($\mathbf{I}$) is sampled according to a sample map ($\mathbf{S}$) to product the scanning pixel camera image ($\hat{\mathbf{I}}$). The target sample rate (κ_a), is selected by the user prior to inference. $\hat{\mathbf{I}}$ is then passed through a feature encoder module ($\mathcal{E}$) parametrised by weights ($\mathbf{w}$) to produce shared intermediate feature representation ($\mathbf{F}$)

$$\mathbf{F}_t = \mathcal{E}(\mathbf{I}_t \odot \mathbf{S}_t | \mathbf{w}_{\mathcal{E}}) \tag{4.1}$$

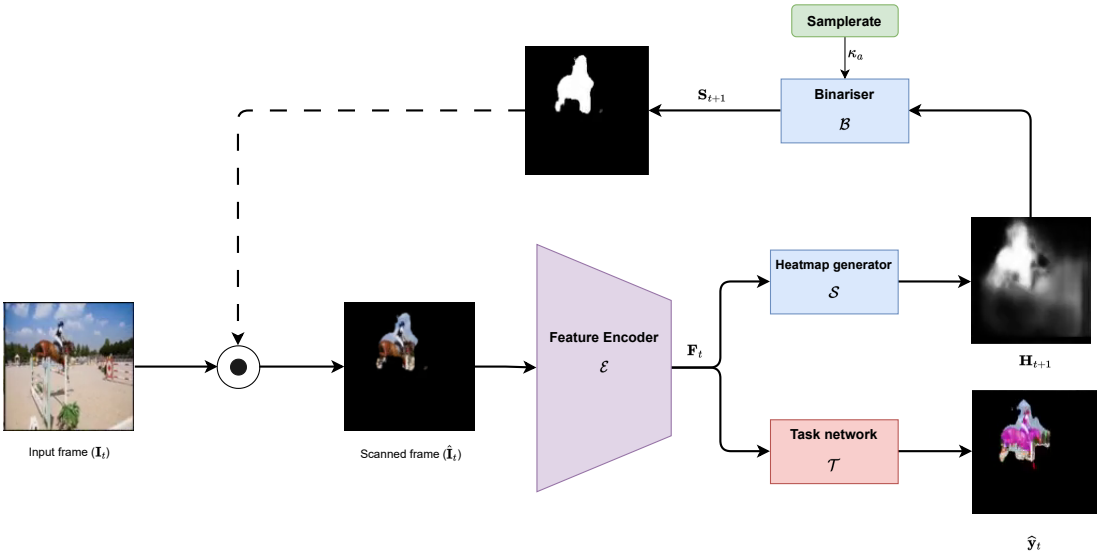


Figure 4.2: Overall architecture, showing the main components for the processing of one frame for a given task, which in this case is tracking. The input scene is scanned according to the sample map ($\mathbf{S}$) generated on the previous frame. The resulting scanning pixel camera image ($\hat{\mathbf{I}}$) is fed into a feature encoder which produces features for both the task and the sample map generator. This produces the sample map for the next frame.

The intermediate feature representation will be used both by the downstream computer vision task ($\mathcal{T}$) and to generate sample maps at future frames,

$$\hat{\mathbf{y}}_t = \mathcal{T}(\mathbf{F}_t, \mathbf{w}_{\mathcal{T}}) \tag{4.2}$$

$$\mathbf{H}_{t+1} = \mathcal{S}(\mathbf{F}_t, \mathbf{w}_{\mathcal{S}}) \tag{4.3}$$

$$\mathbf{S}_{t+1} = \mathcal{B}\left(\mathbf{H}_{t+1}, \kappa_a\right) \tag{4.4}$$

Here it is important to note that the sampler no longer takes the image as an input, as in chapter 3, rather it takes the features from the feature encoder module. These features will be generated from a previously sampled input and also need to be used by the downstream computer vision task. This means that features will have to contain sufficient information to perform both tasks.

4.1.1 Heatmap generator Design

For the heatmap generator we use a decoder made of residual blocks as shown in fig. 4.3. As a feature encoder module, a pre-trained version of the backbone network found in the downstream

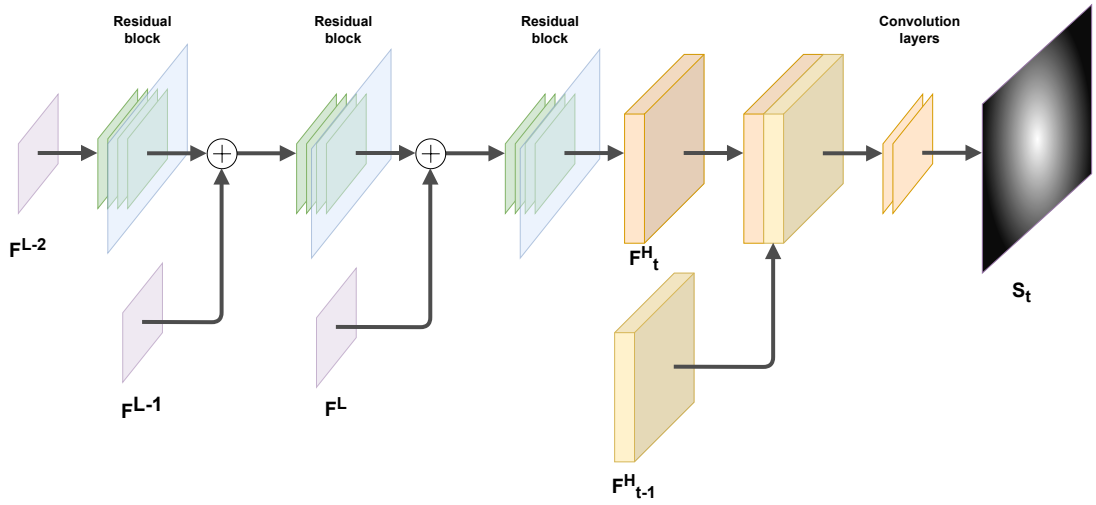


Figure 4.3: The architecture of the heatmap generator. Each of the $\mathbf{F}_t^l$ is sourced from a feature pyramid generated by a feature encoder module, in the same manner as fig. 3.2. Each residual block ends in a $2\times$ upsampling layer in order to ensure that the sizes are correct for the sum operation.

computer vision task is used. This will give a good initialisation for the task, however the task network will still need to be retrained so that the feature encoder module can produce useful features from a sampled input.

The heatmap generator takes the last m feature maps in the feature pyramid of the feature encoder module,

$$\mathbf{F}_t^l \text{ where } l \in [(L-m)..L] \tag{4.5}$$

The value of m, as well as the sizes of the intermediate feature representations, will vary based on the task network used. An additional resize layer is added on the end if the size of the resulting sample heatmap does not match that of the input frame. For the experiments in this chapter, the sizes of $\mathbf{F}^l$ and m can be found in appendix D

Each $\mathbf{F}_t^{l-1}$ of these is passed through a residual block ($\mathcal{R}$) and resized using sub-pixel convolutions before being summed with the the next set of features, $\mathbf{F}_t^l$.

$$\mathbf{F}_t^l = \mathcal{R}\left(\mathbf{F}_t^{l-1} + \mathbf{F}_t^l\right) \tag{4.6}$$

Once all of the input features have been added, we end up with the final sample heatmap features, $\mathbf{F}_t^L$. In order to encode temporal information, the $\mathbf{F}_t^L$ are concatenated (as represented by the $\oplus$

operator) with the previous frame's heatmap features, $\mathbf{F}_{t-1}^{L}$ before passing these through two final convolution layers to produce a sample heatmap ($\mathbf{H}$) logits.

$$\hat{\mathbf{H}}_t = \mathrm{Conv}(\mathbf{F}_t^L \oplus \mathbf{F}_{t-1}^L) \tag{4.7}$$

After this step has been completed the sample heatmap will then be passed through the binariser described in section 3.1.2 to form $\mathbf{S}$ for the next frame.

4.1.2 Losses and Training

To train this framework to use the target sample rate (κ_a) eq. (3.11) is used. The target sample rate is chosen according to the scheduling described in section 3.1.4. The task network is trained using its associated loss. This encourages the sampler and the task to be able to operate on partially sampled images. However this time they are applied to different network heads rather than at different stages during the network, and have equal weighting.

In theory, it is possible to directly train the heatmap generator using the gradient from the feature encoder module. However, doing this would result in a large back-propagation chain, as the gradient would have to flow through the network once for each frame. This would require a significant amount of compute to preform and furthermore risks vanishing/exploding gradients that would render the training ineffective.

To overcome this problem, the sample map generator is trained using a task related ground truth mask ($\mathbf{G}$), such as the ground truth tracking mask is used as the ground truth. This is done so that the sample map generates samples that are specific to the task at hand. For example a sampler trained for human action recognition may pay more attention to multiple foreground objects involved in the interaction, while a tracker may focus primarily on its target. In the absence of such a ground truth it can be possible to generate these using visual explanation techniques such as saliency estimation. Section 3.3 shows that this can be effective when training such models, particularly when it is clear that there is a salient object in the scene.

This allows the training to be broken up into frame sized steps, which is is significantly faster to process owing to the shortening of the back-propagation chain. Once the source of the task related ground truth mask has been decided, the heatmap generator is trained using a binary

cross-entropy loss.

$$\mathcal{L}_{\mathcal{S}} = \mathbf{G}_t \log(\hat{\mathbf{H}}_t) + (1 - \mathbf{G}_t) \log(1 - \hat{\mathbf{H}}_t) \tag{4.8}$$

4.1.3 Initialisation and reinitialisation

The initial sample map for each sequence is a full frame of samples

$$\mathbf{S}_0 = \mathbf{1} \tag{4.9}$$

This gives the sample map generator the opportunity to detect areas of interest for sampling in subsequent frames of the scene. However, note that the effectiveness of the sampler depends to some extent on the effectiveness of the sampling at the previous frame. This iterative relationship ensures that every sampler will eventually fail over long enough periods. To mitigate this we introduce a tracking style re-initialization module where the frame is once again fully sampled if the number of samples in the sample map ($\mathbf{S}$) is smaller than 5% of the maximum number of samples that can be taken from a scene (N). In the case where we have a target mask (for example, in tracking), it is better to use the number of pixels in the initial target mask if this is smaller than the 5% threshold. This threshold can be determined from the results in chapter 3, as even with the best choice of samples, accuracy would drop to below 50% of the peak accuracy below this sampling rate.

Algorithm 2 sample map reinitialisation

procedure REINITIALISE($\mathbf{S}$, N)
  if $\frac{1}{N}\sum \mathbf{S} \leq 0.05$ **then**
    $\mathbf{S} \leftarrow \mathbf{1}$
  else
    $\mathbf{S} \leftarrow \mathbf{S}$
  end if
  Return $\mathbf{S}$
end procedure

4.2 Evaluation protocol

4.2.1 Training

All aspects of the network are train using the ADAM optimiser [31] with a learning rate of 0.0001, with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. All models were trained until convergence. An exponential decay of 0.98 was applied to the learning rate. Further training parameters can be found in the appendix.

4.2.2 Metrics

As with chapter 3, the performance of the model is evaluated across a range of sample rates, plotting the resulting output performance against the sample rate that the model achieves. Ideally, the system should again achieve high task performance across all sampling rates, especially the more challenging low sampling rates where higher data compression is achieved.

In all evaluations the upper bound performance is shown, where the task network is run on fully sampled images. We compare our results to regularly downsampling and uniform randomly sampling the input. Also compared is the Mixed Adaptive Random sampling [69] of the previous frame. There are notably no modern methods able to perform this task. This is because they either rely on future information in order to suggest sampling or only operate on a fully sampled frame [23, 43].

To test for sampling bias (e.g. the object of interest is always in the centre) a version of our network that has uniformly distributed input features instead of using the task features is trained for comparison This referred to as the learned bias.

The resulting metrics are summarised in table 4.1, which shows that on average our method has 11% better performance than the next best method, and only 4% worse than the upper bound. Also shown are some qualitative results in the following sections, along with more detailed performance breakdowns.

	SSM	STM	TransPose	Relative
Upper Bound	0.86	0.80	0.75	1.00
Downsample	0.70	0.62	0.62	0.84
Uniform Random	0.70	0.64	0.51	0.81
Mixed Adaptive Random	0.66	0.67	0.47	0.78
Learned Bias	0.76	0.63	0.58	0.85
Ours	**0.84**	**0.75**	**0.72**	**0.96**

Table 4.1: Absolute and relative performance of all samplers, given as area under curve of each of the task metrics. Relative is computed by comparing the absolute performance to the value of the upper bound, then averaged over all models. The SSM and STM models are averaged over both DAVIS and YouTube-VOS datasets. SSM and STM use mean Jaccard index while TransPose uses PCK, both of which give an accuracy metric between 0 and 1.

4.3 Visual object segmentation evaluation

Visual object segmentation attempts to find a predefined target object in frames of a video. The target object is usually defined in the first frame, by a user or a separate automatic process. This is very similar to object tracking, however rather than defining a bounding box the object is represented by a segmentation, allowing for a more detailed definition of where the object is within a frame. Therefore, each task prediction ($\hat{\mathbf{y}}$) is formed from the current frame ($\hat{\mathbf{I}}^t$), the first frame ($\hat{\mathbf{I}}^0$) and the ground truth for the first frame ($\mathbf{y}^0$)

$$\hat{\mathbf{y}}^{t+1} = \mathcal{T}\left(\hat{\mathbf{I}}^t, \hat{\mathbf{I}}^0, \mathbf{y}^0\right) \tag{4.10}$$

As a result of the similarity, many deep learning tracking techniques apply to visual segmentation. These are usually variations of the basic form as described in [4], where a pair of neural networks that share weights provide intermediate feature representations of both the frame and the target object. These are then combined and a network head is used to determine where the object actually is in the frame. For this task the combined features will be used as intermediate feature representation ($\mathbf{F}$) for the sampling model as these contain information on both the scene and what needs to be looked at in the scene.

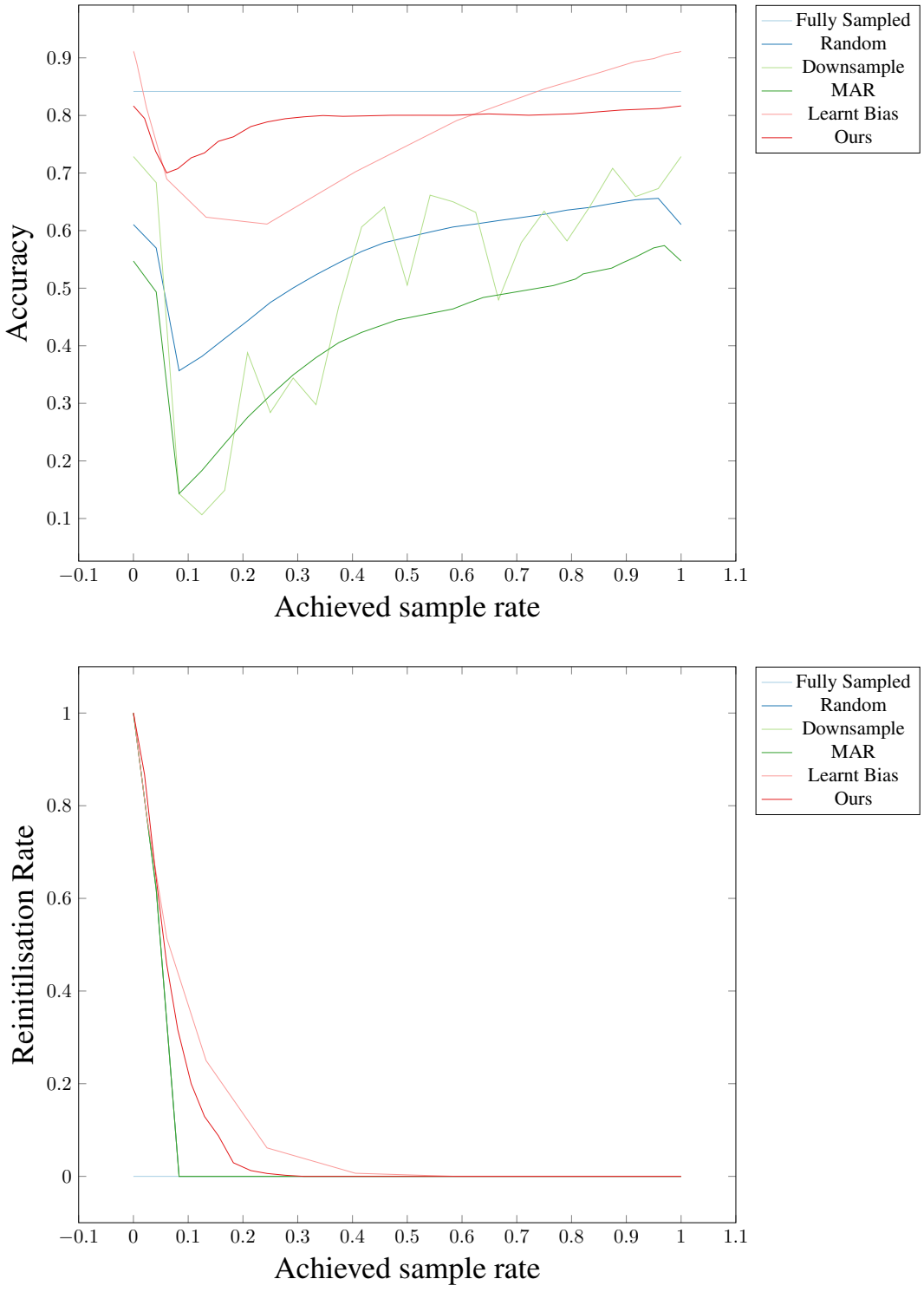


Figure 4.4: Tracking accuracy of SSM-VOS on the DAVIS dataset.

4.3.1 Experimental setup

Testing of visual object segmentation will be done on the DAVIS dataset [47]. The experiment is replicated on the much larger YouTube-VOS dataset [68]. For this task, the success metric is defined as the mean Jaccard index between the target and predicted masks.

For each task, it is useful to establish a usable lower bound for the sample rate, so again results for all sample rates are displayed. The sample rate is measured before any reinitialisation takes place, so also presented are the reinitialisation rate to show the rate of failures.

As downstream computer vision tasks pretrained versions of SSM-VOS [77] and STM [44] are used. These work by examining features in a user selected reference mask and comparing them to features in the current frame. SSM-VOS does this by leveraging similarities between the reference and the current frame at a structural level. In contrast, STM uses a memory network to access a compact representation of all previous frames, allowing access to more information to decide where the tracking target is.

4.3.2 DAVIS

Results for this are shown in figs. 4.4 and 4.5. These clearly show that our sampling method outperforms the other methods consistently. The method presented in this chapter is able to maintain an accuracy of above 80% up to a 20% sample rate, after which the reinitialisation begins to artificially boost the accuracy. For SSM note that the next best is the sampling learned bias. This is unsurprising as in tracking the object is likely to be in the centre of the image. Our method has a slightly higher reinitialisation rate as it is able to set a number of samples that is not equivalent to the target droprate. Some qualitative results are shown in fig. 4.6, taken at a target retention rate of 0.25.

The qualitative results show that the tracker has good performance as long as our sampler is over the tracking target. Areas of the target that are not tracked tend to be outside the sampling. Also, the sampling allocates some importance to the target's immediate surroundings, which allows the task network to have some information about the target's context.

We note that other samplers are not able to achieve the upper bound of performance even when sampling all pixels. This is due to that fact that the trackers are optimised jointly with the

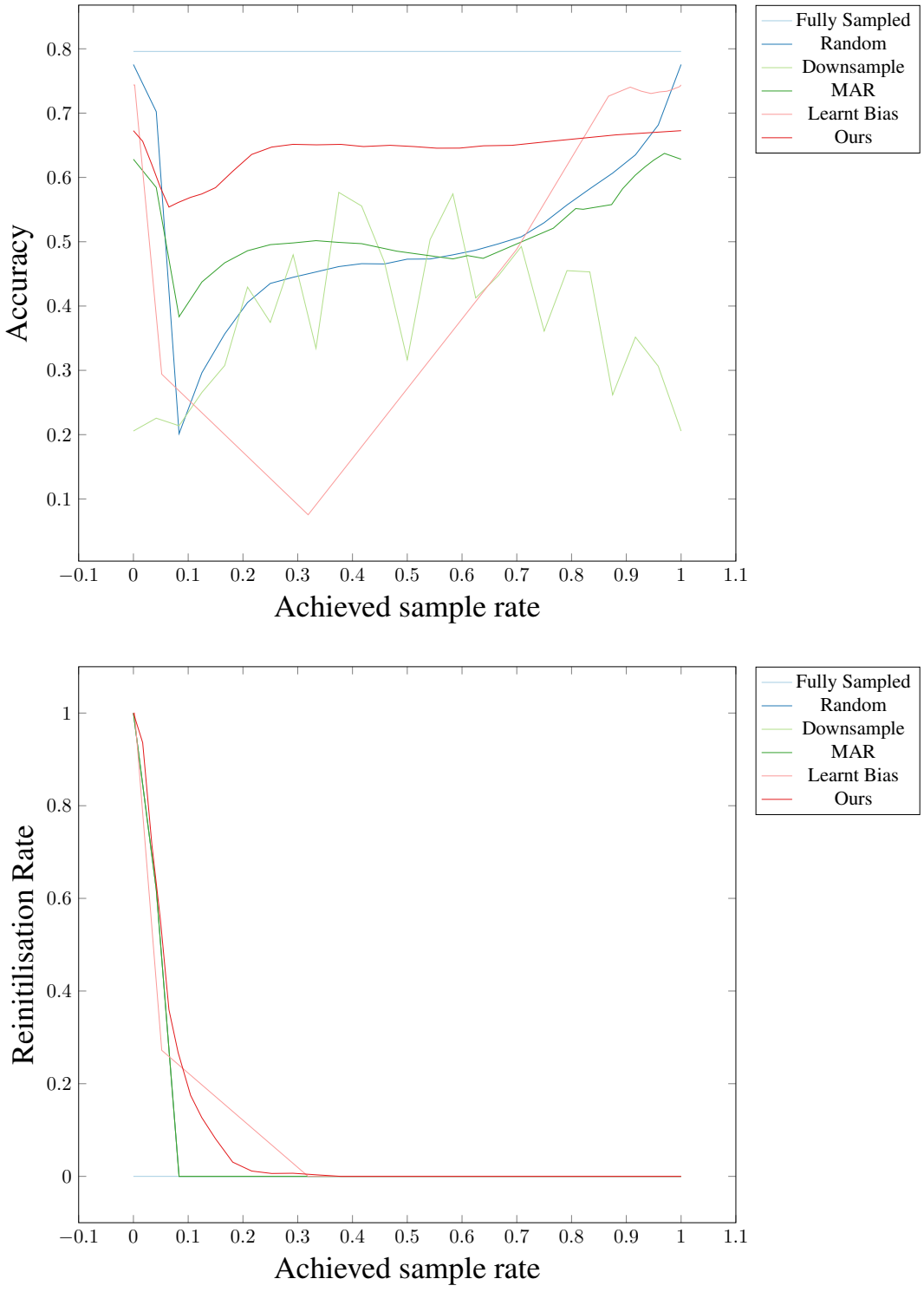


Figure 4.5: Tracking accuracy of STM on the DAVIS dataset.

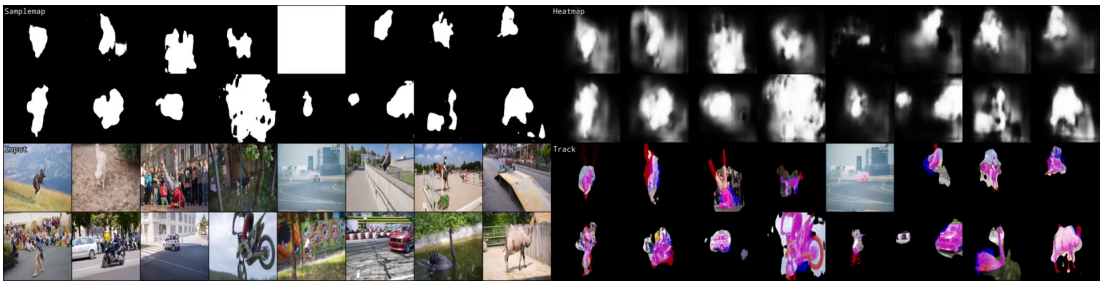


Figure 4.6: SSM tracker on the DAVIS dataset. Top row shows the predicted samplemaps (left) and their associated heatmaps (right). Bottom row shows input frame (left) and tracking result (right), with target highlighted in red and prediction in blue.

samplers so have to learn to deal with fully sampled and sparsely sampled images, resulting in a degradation of performance at all levels. In contrast, our method provides only relevant samples, making it easier for the tracker to produce a result at any sample rate.

4.3.3 YouTube-VOS

Figures 4.7 and 4.8 show our results for SSM and STM respectively. Both trackers have a reduced performance, particularly STM, our method still produces better results. For both task models the learned bias is able to outperform our proposed techniques at the highest sample rates and for STM downsample displays comparable performance above a 50% sample rate. However, the performance of these techniques drops off significantly at the more valuable lower sampling rates.

4.4 Pose Estimation Evaluation

For this task, the aim is to establish the positions of the joints of a person in an image of scene. These can be mapped to a skeleton to provide useful information for tasks such as gesture and action recognition. In this case, the inputs are simpler than the visual object segmentation,

$$\hat{\mathbf{y}}^t = \mathcal{T}\left(\hat{\mathbf{I}}^t\right) \tag{4.11}$$

Early deep learning based methods such as DeepPose [58] regressed the co-ordinates directly, however most modern approaches predict a heatmap for each individual joint [65, 56, 45]. This

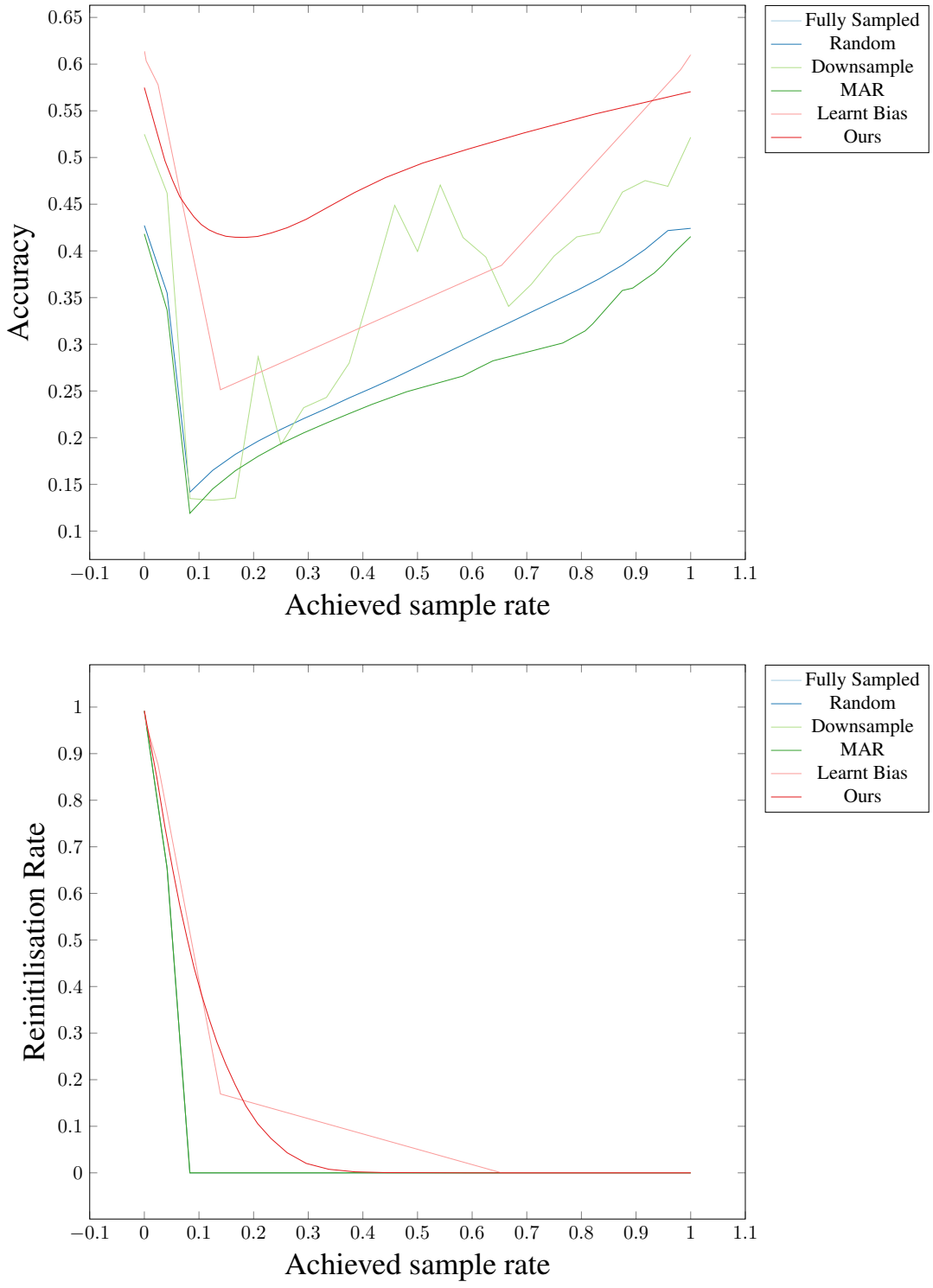


Figure 4.7: Tracking accuracy of SSM on the YouTube-VOS dataset.

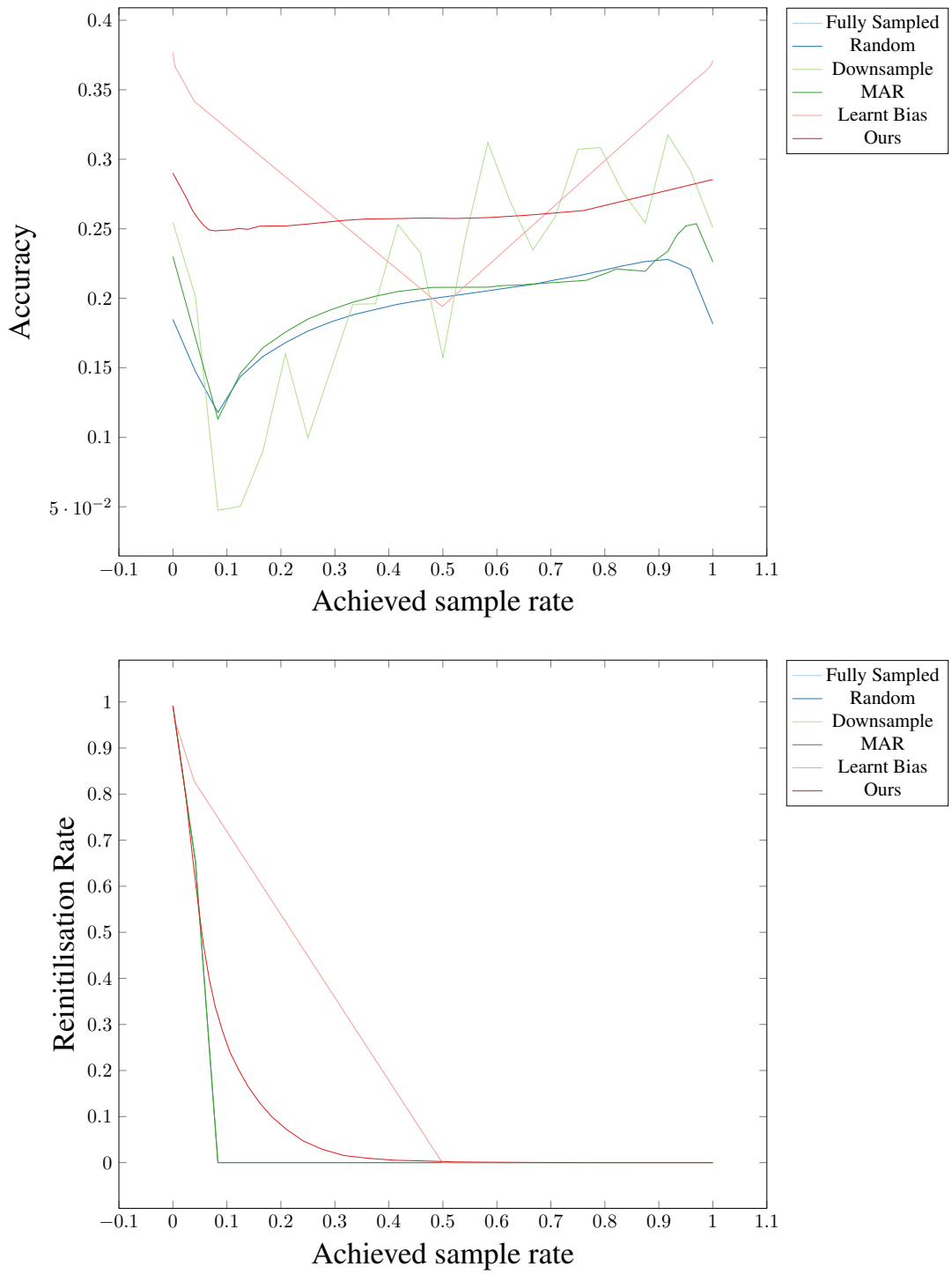


Figure 4.8: Tracking accuracy of STM on the YouTube-VOS dataset.

allows for better localisation of the joints as the heatmap provides a stronger gradient to train the network with.

4.4.1 Experimental Setup

For this task the TransPose network [70] is used to predict joint positions in the JHMDB dataset [30]. TransPose uses transformer layers to produce heatmaps for each of the individual joints, allowing it to maintain a global spatial coherence between each of the joints. The output of these TransPose is designed to operate on individual images, so when applying it to a sequence some degradation in the coherence of the output is to be expected. For this task, the success metric is defined as the mean PCK [71].

4.4.2 Pose estimation results

Despite being a radically different task, fig. 4.9 shows that our method still works well and is able to significantly outperform any competing sampler. This is particularly notable at lower sampling rates ($< 30\%$). This is likely due to the fact that a person is likely to take up a relatively small area of a frame, so methods that provide sparse samples globally are not likely to pick up on the relevant details for joint position estimation. Additionally, the learned bias is much less effective here due to the lack of a strong central bias in this dataset when compared with tracking datasets.

Qualitative results in fig. 4.10 show that the sample maps are much more closely targeted to the person in question than they were for the tracking application. This is likely due to the lack of temporal information that will be present in the features of the TransPose task network.

4.5 Achieving the target sample rate

It is important to know how well each model achieves the target sample rate as it is necessary to evaluate what a good value of target sample rate (κ_a) would be to use in a real world situation. To evaluate this the desired target sample rate is compared to the sample rate achieved by the

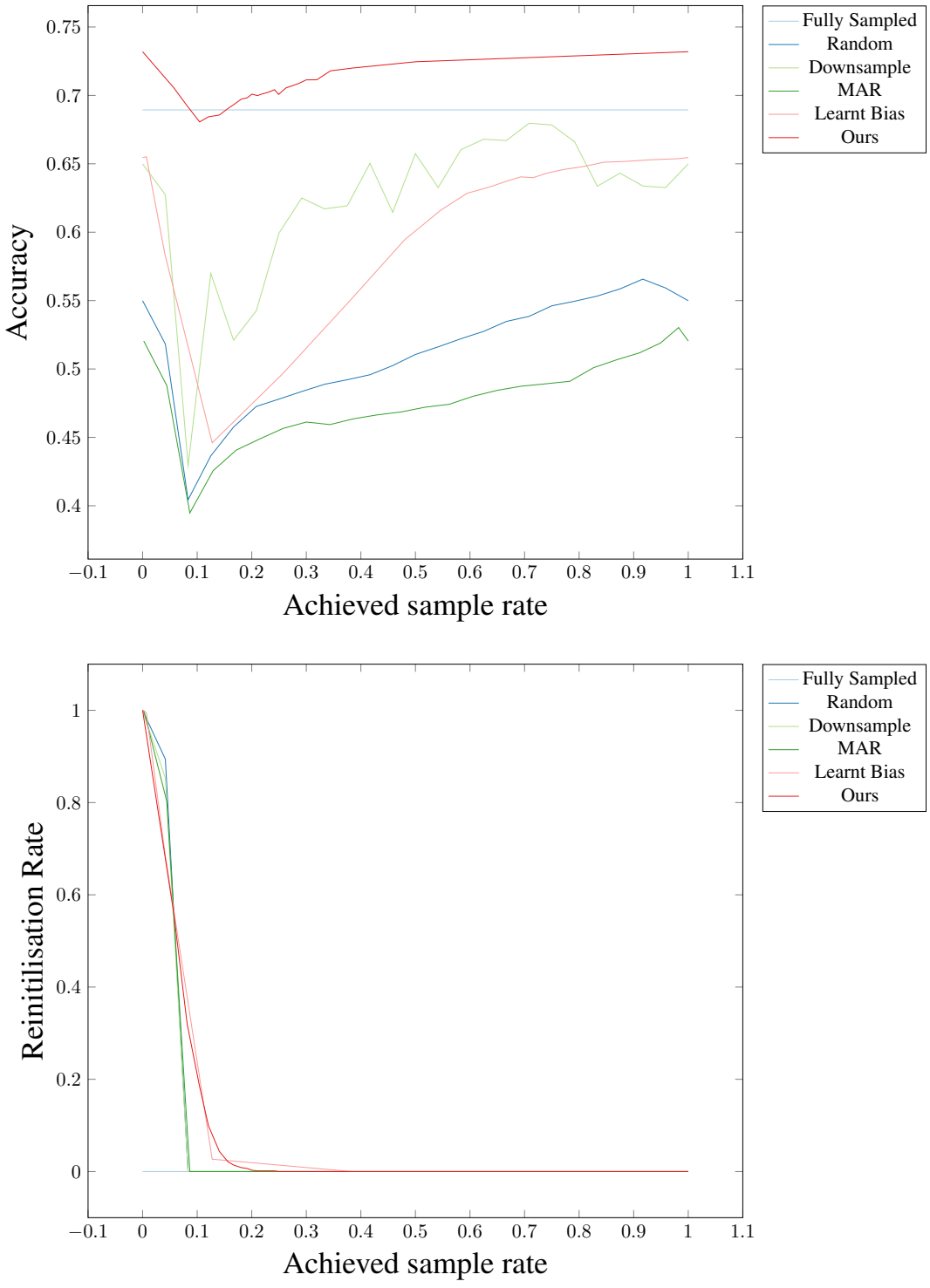


Figure 4.9: Accuracy of TransPose on the JHMDB dataset.

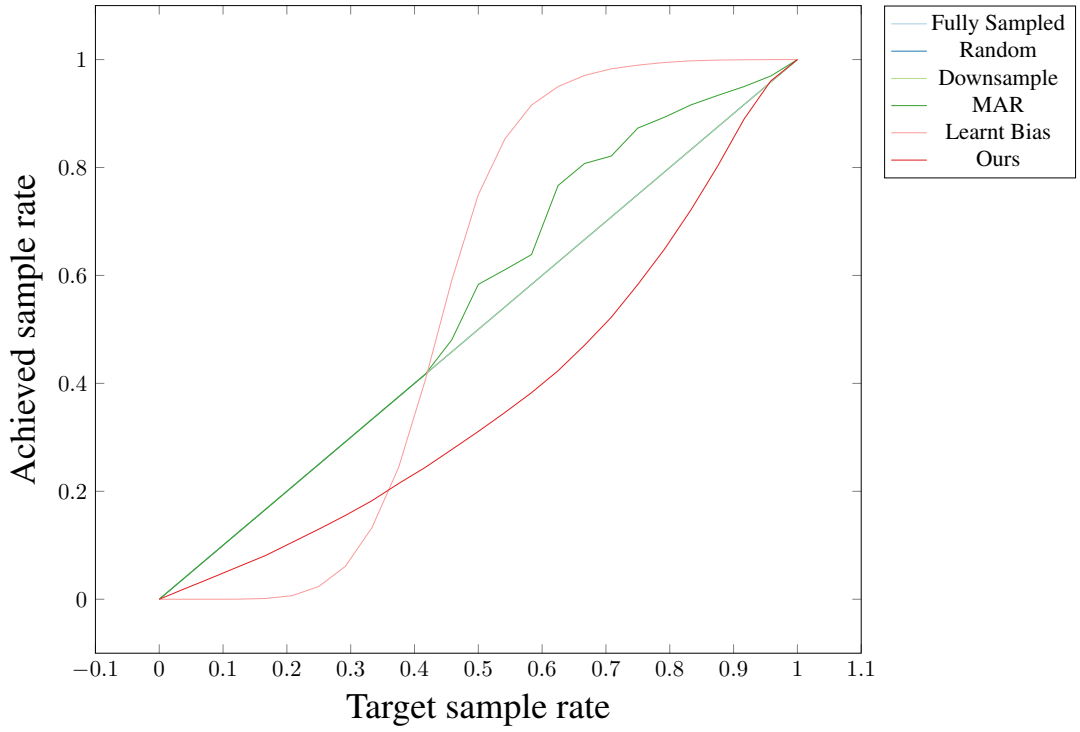
Figure 4.10: Qualitative results for our method using TransPose. Top row shows input frame (left) and joint estimation result (right), with target highlighted in red and prediction in blue. Middle row shows the predicted samplemaps (left) and their associated heatmaps (right). Bottom row shows the sampled input frame (left) and the overlaid pose estimation result (right).

model before any re-initialisation. The results of this for each model are shown in fig. 4.11. The results for SSM and STM models are displayed for the DAVIS dataset.

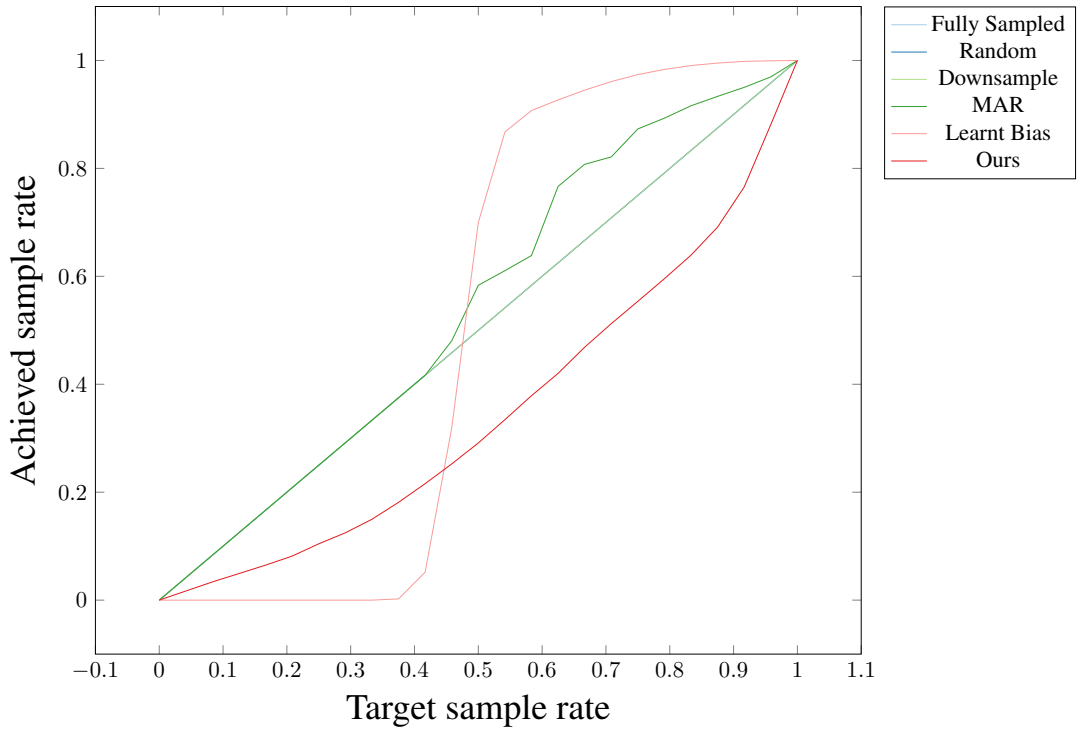
While downsampling or uniformly randomly sampling the input will always achieve the target sample rate, learned sampling approaches are not guaranteed to. Figure 4.11 shows that our model tends to underestimate the target retention rate. However, in practice this is not a problem as there exists an easily definable mapping between the achieved retention rate and the desired retention rate. In the bias detector there is more of a problem as while the mapping still exists the centre is flatter so is more difficult to achieve reliably.

4.6 Saliency based training

As established in section 3.3, it is possible to train the heatmap generator using saliency maps. This is particularly useful for this application as not all downstream computer vision tasks will provide a ground truth mask. Furthermore, as the tasks presented in this chapter are localised to particular areas of the image, this is a prime candidate for saliency based training to be effective.



(a) SSM



(b) STM

Figure 4.11: How well each method achieves the target droprate across all models

To test this holds in the predictive case the experiment in section 4.3.2 can be repeated. However, rather than using the task ground truth as task related ground truth mask ($\mathbf{G}$) to train the heatmap generator, instead a saliency prediction network is used to generate a $\mathbf{G}$ from the image ($\mathbf{I}$). This is then tested this using the SSM network for visual object segmentation of the DAVIS dataset, with the results shown in fig. 3.9.

These results still demonstrate the superiority of the technique suggested within this chapter. However, the difference is that the reinitialisation kicks in earlier than when trained with the pure ground truth masks. This is particularly significant for the learnt bias, where the re-initialisation significantly increases at a 50% target sample rate. This is also true for our method, with re-initialisation starting at 37% rather than 20% for the same task. This makes the effective sample rate higher.

4.7 Conclusion

In this chapter a general framework applicable to any video based network has been developed. This allows for the user to trade between output accuracy and computational complexity. It is able to generalise to a variety of tasks and can be easily used by a number of different models. The value of this approach has been demonstrated across multiple target tasks, multiple datasets and different task specific models. Overall, the proposed method are able to achieve comparable performance to the current state-of-the-art on these problems while reducing storage, transmission and processing complexity by 80%. This potential efficiency gain may benefit deploying future deep learning algorithms on constrained hardware such as in the space sector and internet of things devices. More fundamentally, this reduction in data transmission and processing may also have positive implications for the environmental impact of the AI revolution.

This is an important step in realising the advantages of a scanning pixel camera as we can now how have a fully predictive sampling setup. However, this method still has to sample a frame's worth of samples at a time before the task network is executed. This is an unnecessary constraint on a scanning pixel camera, and the next chapter will seek to remove this constraint as well, bringing the realisation of a scanning pixel camera ever closer.

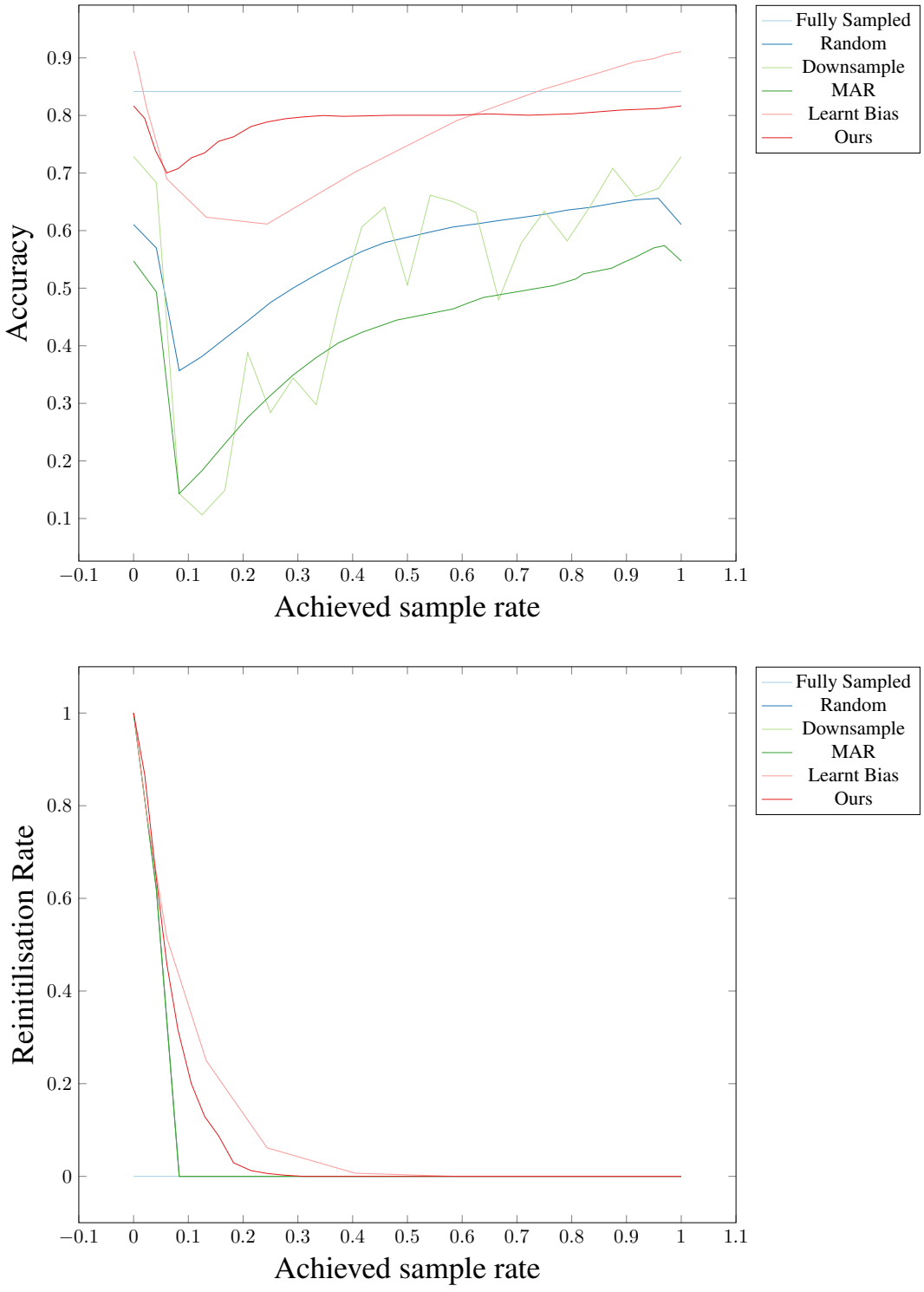


Figure 4.12: Saliency based training results for visual object segmentation

Chapter 5

Continuous Sampling

So far this thesis has assumed that samples are taken a frame at a time. However, a scanning pixel camera does not have to be bound by this paradigm, especially when it comes to more dynamic scenes. In fact, only sampling a frame's worth of pixels at a time is not fully leveraging the capabilities of a scanning pixel camera. As each new collection of samples brings in new information about the scene, it should be possible to simultaneously update both the task prediction and decide where best to sample next to ensure that this task prediction is accurate. By following this paradigm, all the simplifications made in section 1.4 can be removed, treating the scanning pixel camera as a device that produces a continuous stream of samples, rather than a synchronised set of pixels.

5.1 Methodology

The overall framework for the method in this chapter is shown in fig. 5.1. It bears similarity to the method in the previous chapter however there are some key differences. Firstly, samples are generated iteratively in a sequence, for a single static scene. Secondly, the task ground truth ($\mathbf{y}$) will be the same for each sample, and the prediction should become more accurate as more samples are taken for the scene.

As with previous chapters, it is desirable to minimise the number of samples taken from a scene (n) that are taken in order to complete the task. However, rather than matching a target

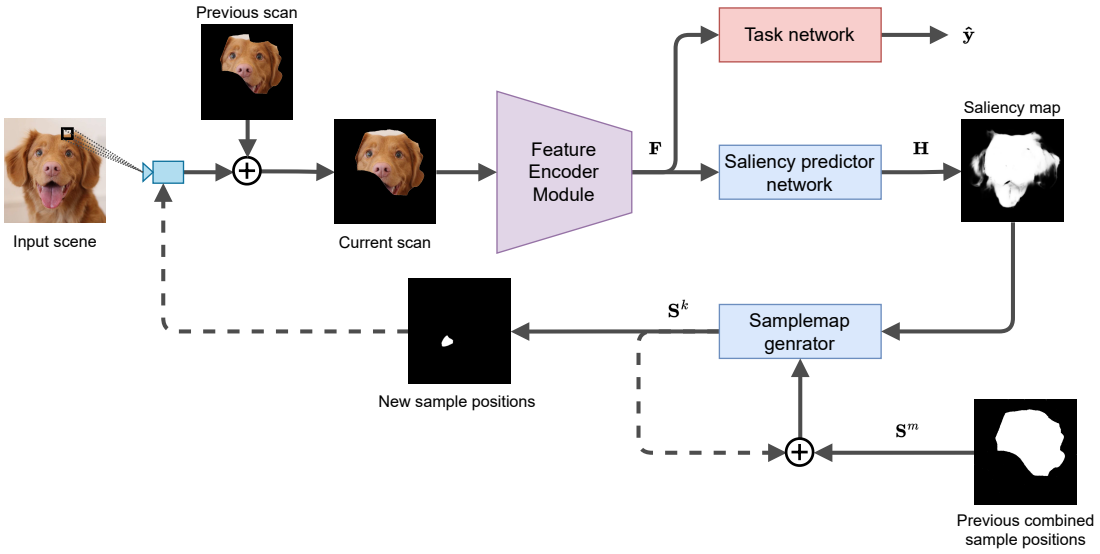


Figure 5.1: New samples are accumulated into a scan, as denoted by the $\oplus$ operator. This creates a current scan that is used as an input to the task via a feature encoder. The intermediate feature representation ($\mathbf{F}$) are also used to predict a saliency map, which is then used to generate the next sample positions to be accumulated into the scan.

sample rate, this chapter seeks to minimise the sample rate completely, which applies the following constraint to n,

$$\min n \mid n \in [0..N] \tag{5.1}$$

This technique is to be applied to any downstream computer vision task, assuming that the task network contains a feature encoder module ($\mathcal{E}$) which provides intermediate feature representation ($\mathbf{F}$). These intermediate feature representations should contain information about the salient information of the scene w.r.t. the task they were trained for, therefore they can be used to generate useful sample maps for future samples. The downstream computer vision task will be evaluated every time a new sample is acquired.

5.2 Selecting new samples

The feature encoder module produces an intermediate feature representation from an incomplete scanning pixel camera image ($\hat{\mathbf{I}}$) with k samples in positions defined by the sample map $\mathbf{S}^k$. These features are used to generate a task prediction for that amount of sampling ($\hat{\mathbf{y}}^k$). The

intermediate feature representation is also used to predict a sample heatmap to place the next m samples ($\mathbf{H}^m$) in order to best complete the task . m samples are used rather than a single one as a single sample may not provide enough new information to significantly alter the task prediction.

This is done using a heatmap generator ($\mathcal{S}$) that outputs $\mathbf{H}^m \in [0..1]$ as follows,

$$\mathbf{H}^m = \mathcal{S}(\mathbf{F}^k) - \mathbf{S}^k \tag{5.2}$$

The subtraction of the previous sample map guarantees that $\mathbf{H}^m$ will be zero at positions that have already been sampled, and therefore will not be resampled. The sample heatmap is only used at this step and then discarded, so as to prevent bias from previous samples and allow the system to freshly consider all the information in a new context.

The next m samples can then be selected by using maximum likelihood estimation. Any ties are broken at random.

$$\mathbf{S}^m = \max_{\mathbf{A}} \|\mathbf{A} \odot \mathbf{H}^m\| \text{, where } \|\mathbf{A}\| = m, \mathbf{A} \in [0, 1] \tag{5.3}$$

This is different to the heatmap binarisers in previous chapters as sample heatmap ($\mathbf{H}$) no longer needs to be normalised in order to operate with multiple sample rates.

$\mathbf{S}^m$ describes where new samples are to be taken from. These new samples are accumulated with the existing samples and this new scanning pixel camera image is fed into the feature encoder module to begin the process again. Thus the simulated $\hat{\mathbf{I}}$ based on the fully accumulated sample map ($\mathbf{S}$) can be defined as,

$$\hat{\mathbf{I}}^{k+m} = \mathbf{I} \odot \left(\mathbf{S}^k + \mathbf{S}^m\right) \tag{5.4}$$

The process described in eqs. (5.2) to (5.4) can then be repeated by feeding the sampled image back into the task network to provide even more samples to further refine the result until all samples have been taken from the scene.

5.3 Sample Saliency estimation

The heatmap generator uses a decoder made of residual blocks as shown in fig. 5.2. The input of each residual block is processed by a transformer in the same manner as fig. 3.3. This is a similar

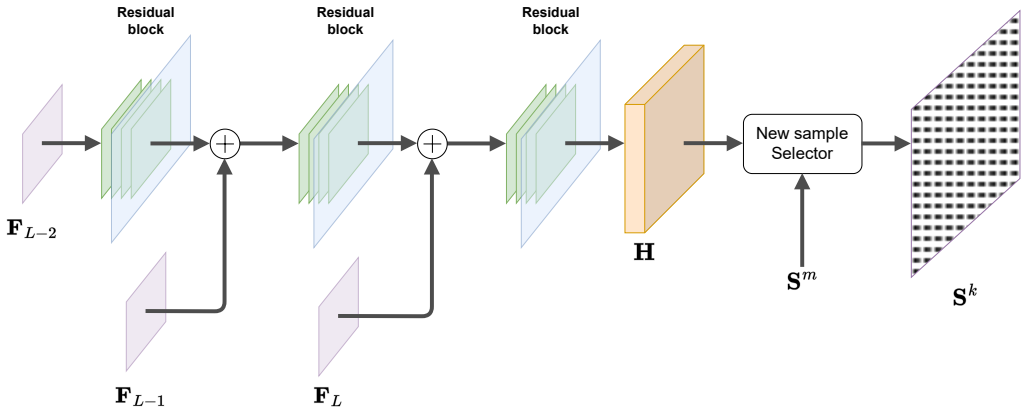


Figure 5.2: The heatmap generator ($\mathcal{S}$) architecture remains similar to previous chapters. Taking a feature pyramid from a feature encoder module ($\mathcal{E}$) to generate an intermediate feature pyramid, each intermediate feature representation ($\mathbf{F}$) from this pyramid is passed through a residual block before upscaling it so that it can match the size of the next set of features. The resulting sample heatmap ($\mathbf{H}$) is then binarised to produce a new selection of samples before being accumulated onto the existing sample map.

architecture to that described in section 4.1.1. It is fed the intermediate feature representation as a feature described by eq. (4.5), as well as the previous sample map ($\mathbf{S}^m$), to produce a sample map ($\mathbf{S}^k$) that represents the new sample positions.

There are a few key changes to the architecture in order to achieve this. Firstly, as mentioned the sample heatmap is no longer normalised as it no longer needs to provide probabilities for all sample rates. Secondly, past frame features are no longer provided as an additional input and instead $\mathbf{H}$ is predicted directly using only the existing set of samples. This is possible as each set of features will be produced using all previous samples, and there is no difference in the content of the respective frames, so the information from previous samples does not need to be retained.

5.3.1 Selecting an amount of samples

Taking a single sample is not likely to yield enough information to meaningfully change the prediction. Thus in order to gain a reasonable amount of information in a single step, it is useful to take m samples in at a time. This is easily supported given the maximum likelihood estimation method used, as we simply have to take the samples with the m highest probabilities.

This presents a different problem, while taking m samples will result in more information, it may result in samples being far away from each other and therefore will provide small amounts of unrelated information. To resolve this rather than pick a single sample m times, a small area of samples is selected m times. These co-located samples provide more information over a small area than a single sample would so are more informative.

This can be done with minimal computational overhead by generating $\mathbf{S}^m$ from a downsampled version of $\mathbf{H}^m$. When this is upsampled using nearest-neighbour interpolation, it results in blocks of samples around each of the desired sampling positions. Therefore, given m new samples and a downsampling scaling factor (ψ), the number of new samples taken at each step ($\hat{m}$) is given as,

$$\hat{m} = \frac{m}{\psi} \tag{5.5}$$

5.3.2 Feature scaling

Initially, the input will be a blank image, which will give a constant scanpattern. This means that the first sample taken will strongly influence where future samples will be taken as it will be the only source of information. This can make the reliability of the system sensitive to the relevance of the earliest samples. In the earlier stages exploration of the scene by the sensor is to be encouraged, rather than exploitation of the currently available information. Depending on the exact scanpattern used, it can be challenging for the feature encoder module to learn to modulate it's behaviour appropriately over time, so instead the intermediate feature representations are scaled based on the number of current samples (k) and the number of samples taken from a scene (n), as follows,

$$\hat{\mathbf{F}}_l = \begin{cases} \frac{k}{n}\mathbf{F}_l & \text{if } \frac{k}{n} \leq 1 \\ \mathbf{F}_l & \text{if } \frac{k}{n} > 1 \end{cases} \tag{5.6}$$

This also has the additional benefit of giving a bias to the first sampling positions, as they will be produced using a completely zeroed input. The initial samples will therefore be placed at the area most likely to give significant information in an arbitrary scene. In other words they represent a prior over important scene locations.

5.3.3 Training procedure

The task network will be trained using the same loss function as the original task network, as with previous chapters. This is done at every sampling step in order to ensure that the network is able to produce results from sampled images.

It is theoretically possible to train the heatmap generator in a recurrent manner, however this leads to vanishing gradient problems and is prohibitively expensive to train. Therefore, a task related ground truth mask is provided to generate optimal sample heatmaps. As it has been sh0wn to be effective in previous chapters, saliency estimation of the can be used to provide an estimate for the task related ground truth mask. Alternatively, for tasks such as segmentation, the task ground truth can be used for this purpose. Whatever the choice, it is referred to as the oracle system ($\mathcal{O}$). Once $\mathcal{O}$ is established it is used as a ground truth in an L1-Loss against the output of the heatmap generator. This also improves robustness and generalisation, as the system is exposed to a broader range of initial sample maps.

$$\mathcal{L}_{\text{salient}} = \|\mathbf{H} - \mathcal{O}(\mathbf{I})\| \tag{5.7}$$

Rather than training on a full sequence of samples for each scene, which would be time consuming, particularly at higher sample rates, the task related ground truth mask ($\mathbf{G}$) can be used to generate a sample map for an arbitrary number of steps. This can then be used to predict a single next step of m samples. By randomly selecting the number of fake previous steps, the entire sequence will be eventually be covered without having to go through the time consuming process of iterating through each individual sample.

5.3.4 Sampling Termination

As established there is a maximum number of samples that can be taken from a scene (N). However, as demonstrated in previous chapters, a significantly lower number of samples taken from a scene can be taken while still maintaining task performance. Therefore, at inference, the sampling repeats until the upper limit of a predefined target sample rate (κ_a), defined in eq. (3.2), is achieved. The information from previous chapters can be utilised to decide on a good value for κ_a.

However, for some inputs, the task network may be able to solve the problem in fewer samples than specified by the sample rate (*e.g.* when the target object takes a relatively small portion of the image). Here open-set recognition techniques are used in order to stop the sampling early, as the stopping point is when the task is confident that task prediction ($\hat{\mathbf{y}}$) is in the closed set. As suggested in [61], the maximum logit score (MLS) is sufficient to determine the stopping condition. There are many other techniques of establishing if the $\hat{\mathbf{y}}$ is in the closed-set, however these are largely dependent on the quality of the model they are applied to, so they are not used in this thesis.

The confidence estimates will be unreliable below a certain proportion of samples (α). Therefore the final stopping condition given a current sample rate (κ) is,

$$(\kappa > \alpha) \ \wedge \ (\text{MLS}(\mathbf{F}_L) > \gamma \ \vee \ \kappa > \kappa_a) \tag{5.8}$$

where γ is the threshold for the maximum logit score. This condition is not applied during training in order to maximise the range of partially sampled images that the system is trained on.

5.4 Classification Evaluation

Firstly, a classification task using the same setup as in chapter 3 is evaluated. This means using EfficientNet-b0 as a classifier and the metric is the classification accuracy. For the intermediate feature representation the last 4 block of the network was used. Details of the sizes for this can be found in [57].

The method presented in section 5.1 is compared to a number of alternative sampling strategies. These include randomly sampling the image and doing a simple raster scan. It is also compared against a learnt bias to ensure our system is exploiting the nature of each individual scene and not simply learning to focus on a general region such as a central image bias. There are a lack of more modern methods to do this task as they either require pre-emptive knowledge of the entire scene.

To test that the base idea works, the classification is first performed on the CUB200 [62] dataset, as shown in fig. 5.3. Initially, a maximum κ_a of 0.5 is chosen. based on knowledge from chapter 3, a Feature Pyramid Attention [75] is used to provide saliency masks to use as an oracle system.

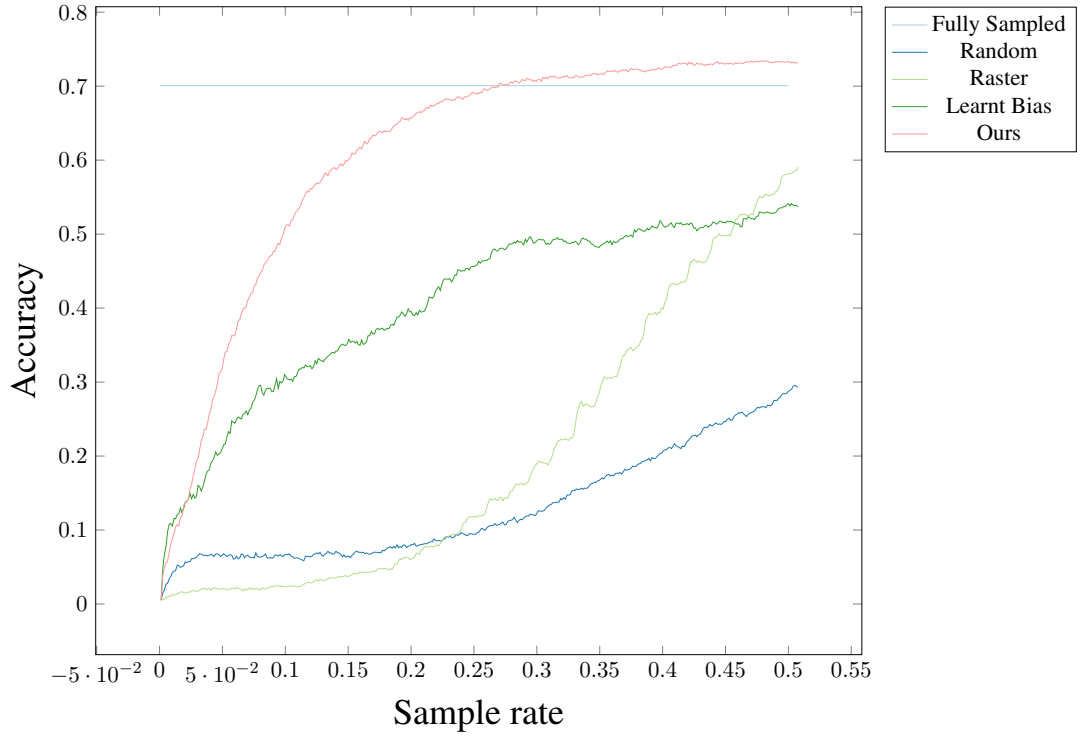


Figure 5.3: Accuracy of classification on CUB-200 dataset

These show that the method significantly outperforms all other comparison models, even at lower sample rates. This confirms that it is picking highly relevant samples right from the very start. Furthermore, our method is able to achieve results above the level of the baseline as it is able to ignore irrelevant information that may distract from the result. Interestingly, it is able to achieve near peak accuracy at a 32% sampling rate despite it being trained with a higher maximum target sample rate (κ_a).

Figures 5.4 and 5.5 show the progression of the heatmaps and their sampling over the course of a scan. This shows that the sampling is well centred on the target bird, which leads to the good results in from the classifier as it only has the relevant information a limited background.

The κ_a that the model is trained at can be lower. Figure 3.9 indicates that it should be possible to go as low as 0.15. However, predicting where to sample without prior knowledge is likely to require more information, so the experiment is repeated with a κ_a of 0.25.

The results in fig. 5.6 are surprising. Although they show that our method is still superior, performance has dropped significantly. It should be possible to achieve comparative accuracies

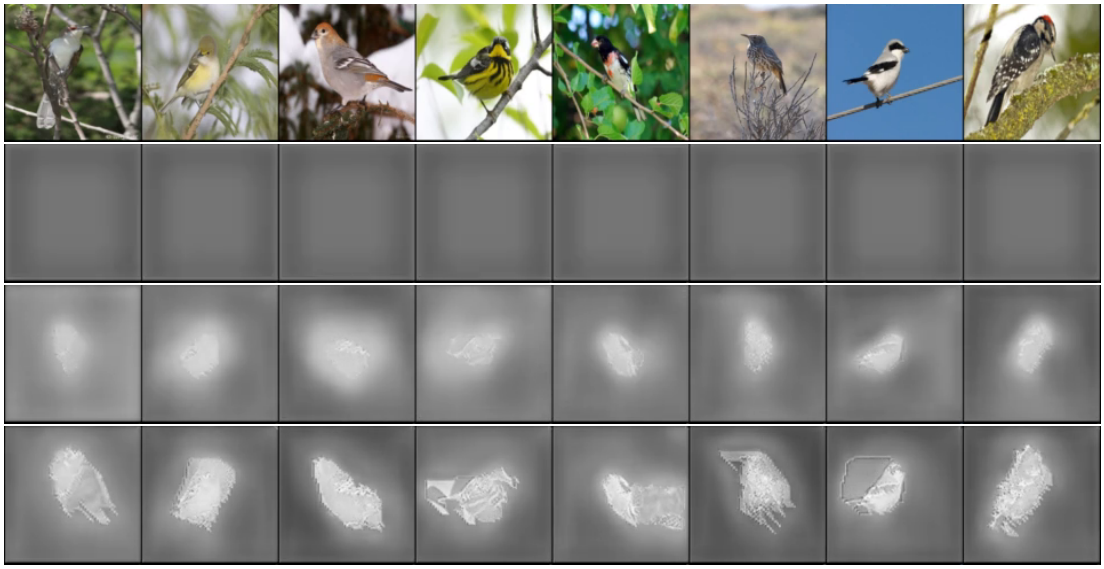


Figure 5.4: Example sampling heatmaps as sampling progresses

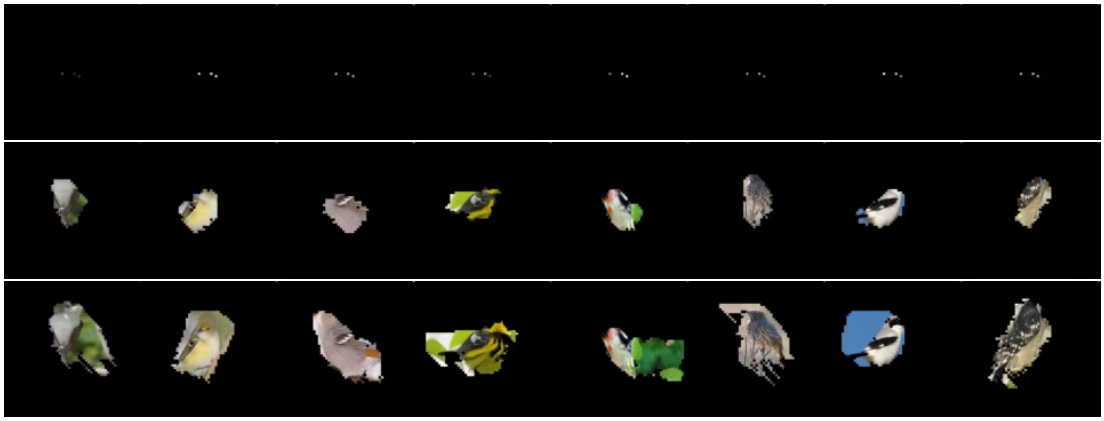


Figure 5.5: Sampling progression examples

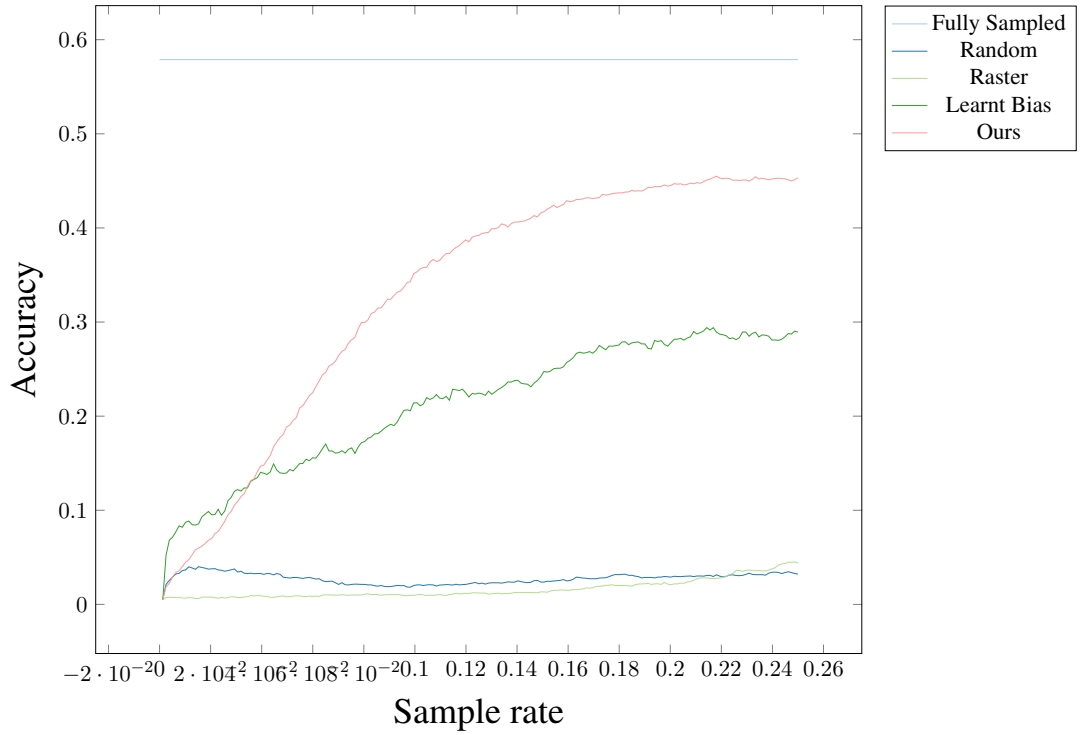


Figure 5.6: Accuracy of classification on CUB-200 dataset at a κ_a of 0.25

using a much lower κ_a. This is likely due to the fact that the heatmap generators in section 3.1.1 are able to see the image in its entirety before placing a sample. therefore, they do not need to waste time exploring a scene to establish where the useful samples are likely to be.

5.4.1 Comparison of different saliency oracles

The choice of oracle used to train the heatmap generator will greatly affect its performance. This is due to the discrepancy between what the oracle chooses to focus on, and what is necessary for the task to make an accurate decision.

In fig. 5.7 the different oracles of DeepSAUCE from section 3.1.1, and a Pyramid Feature Attention Network [75] are compared. It is clear that using the Pyramid Feature Attention Network leads to a significant improvement to the result. This is likely due to the highly discriminative nature of the saliency map directing all the samples into important areas, whereas SAUCE may direct some samples into unimportant areas, due to a base sampling rate even in unimportant regions. DeepSAUCE is able to achieve an overall higher accuracy but does

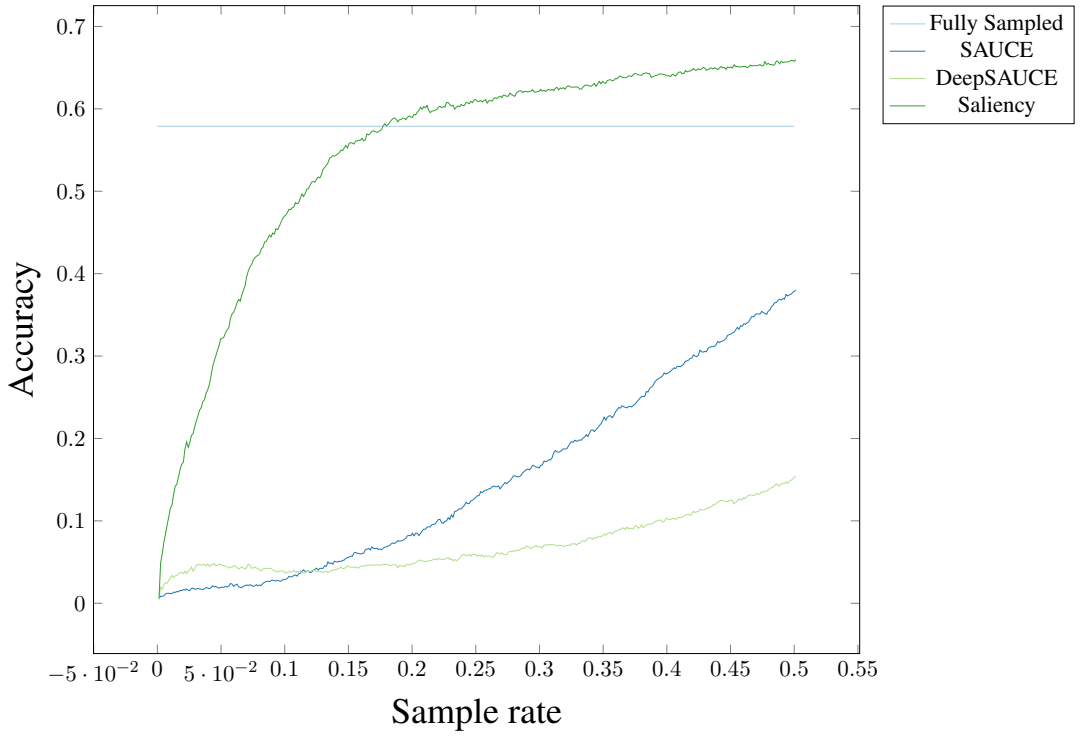


Figure 5.7: Training our method with different oracle systems for classification

so using more samples, so for the experiments in this chapter the Pyramid Feature Attention Network is used as it achieves the desired goal at a much lower sample rate.

5.4.2 Sampling Termination analysis

Figure 5.8 clearly shows that our model leads to much more confident task networks, achieving peak confidence after only 21% of the samples. However, it should be noted that peak confidence does not occur at the same sample rate as peak accuracy, implying that using the peak confidence to stop early would get a less accurate result. This is perhaps unsurprising, as providing additional information rarely worsens a classifiers decision. Nevertheless, it is gratifying to see that an additional increase in efficiency can be achieved via this early stopping.

The method suggested in this chapter is also able to gain confidence much more quickly than randomly sampling and raster scanning. Initially, it has slightly worse performance than the learnt bias before eventually overtaking it. The feature scaling means that at low sample rates it will select samples similar to the learnt bias that are likely to be in salient areas. However,

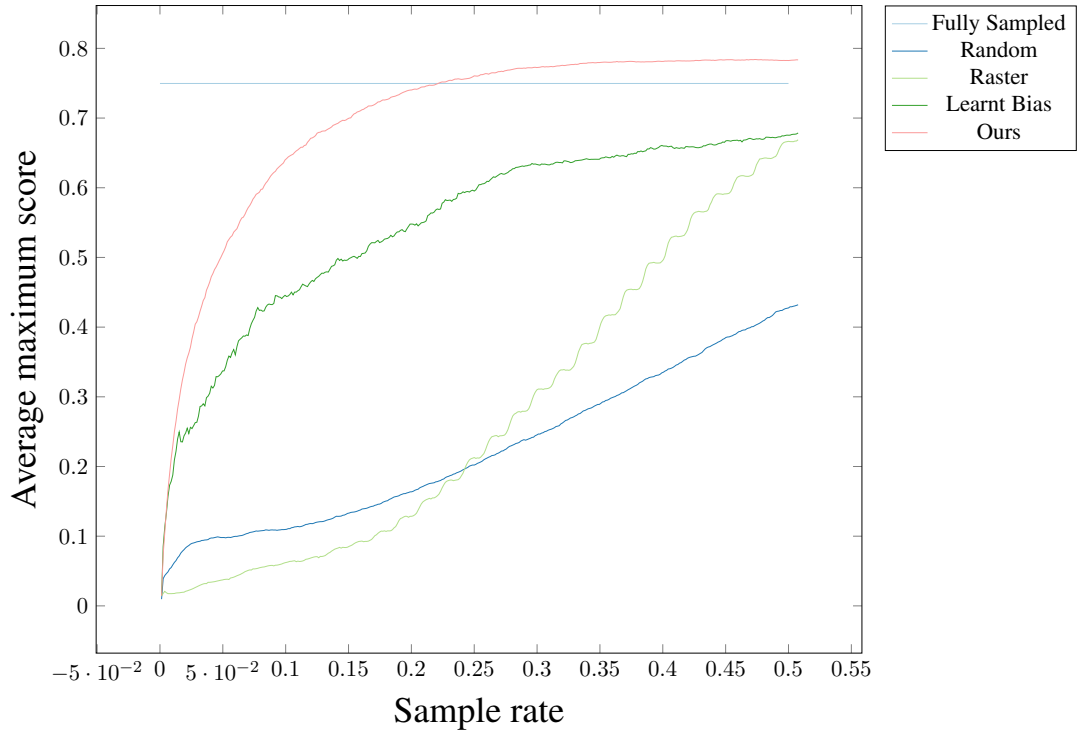


Figure 5.8: Confidence of the task network as sampling increases. Maximum score is calculated as the largest softmax score of the output classes.

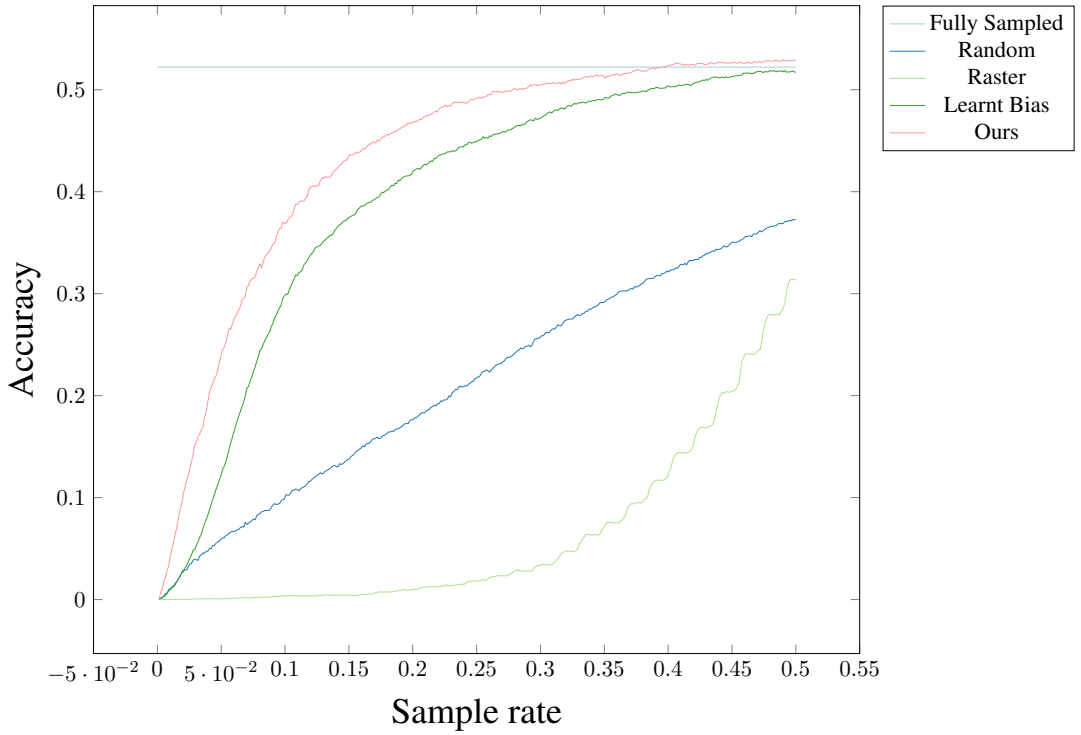


Figure 5.9: Classification of the iNaturalist dataset using EfficientNet-b0

as sampling progresses, our method is able to exploit the information from the samples that it already has and produce more accurate results that it is more confident in.

This can be used to decide a value for γ when using the early stopping condition. A value of 0.7 would give a performance comparable to a fully sampled input using only 20% of the samples. Notably this is now close to the value of 0.15, which was suggested as a lower bound for sampling in fig. 3.9.

5.5 Expanding task coverage

To see how well the method proposed in this chapter generalises, it is tested on other tasks. To expand the classification task, the larger dataset of iNaturalist [28] is used, with the results shown in fig. 5.9. The method proposed in this chapter still outperforms the comparison methods though it takes longer to reach the same performance as a fully sampled input. This is likely due to the fact that iNaturalist presents a wider range of classes so more samples, and consequently more information, must be gathered in order to generate an accurate classification.

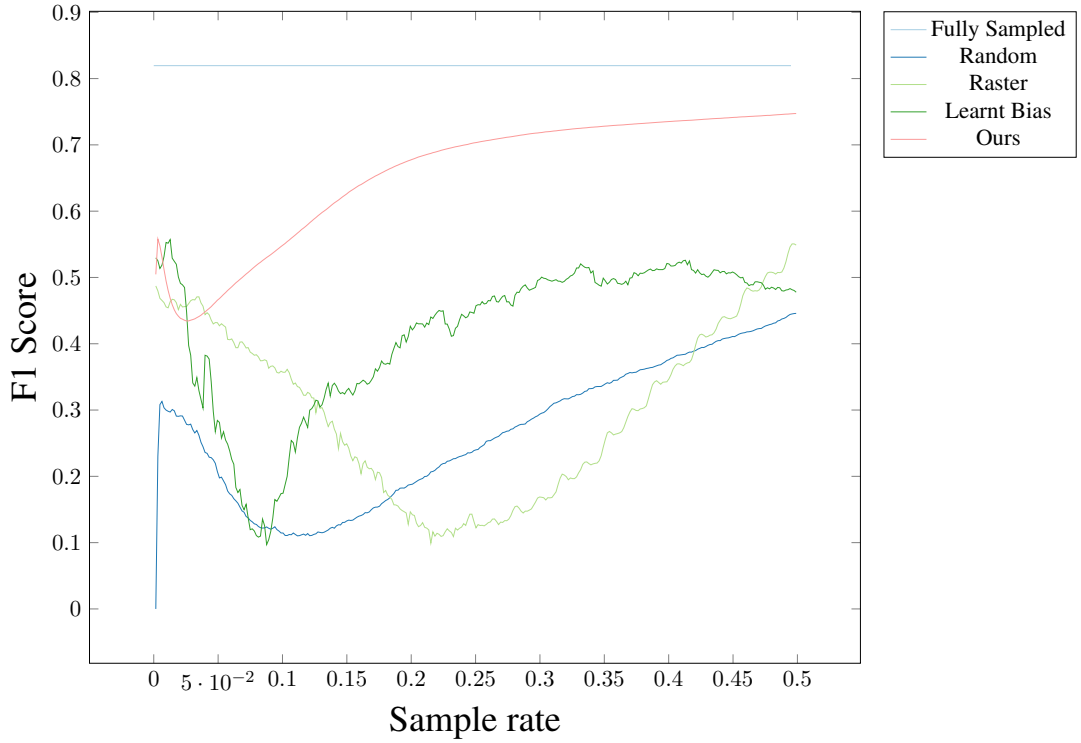


Figure 5.10: CUB200 segmentation using FPN

We also preform segmentation using a Feature Pyramind Network (FPN) on the CUB200 dataset. This used EfficientNet-B0 as an encoder, so the last 4 blocks again were used for the intermediate feature representation. Figure 5.10 shows that our method is more accurate than the comparative methods but is not able to be more accurate than a fully sampled image. This is likely because the task requires more information to form an accurate result, as an exact outline of the target requires knowledge of the edges that will be harder to predict.

An interesting behaviour of this task is the quality of the prediction initially becomes worse in the early stages of sampling. This is likely due to the task network exploiting a bias when using unsampled input. As more samples are taken, the task network is able to produce an image specific prediction. However, if the samples taken do not provide a sufficient amount of information, then the new prediction will be worse than the initial prediction.

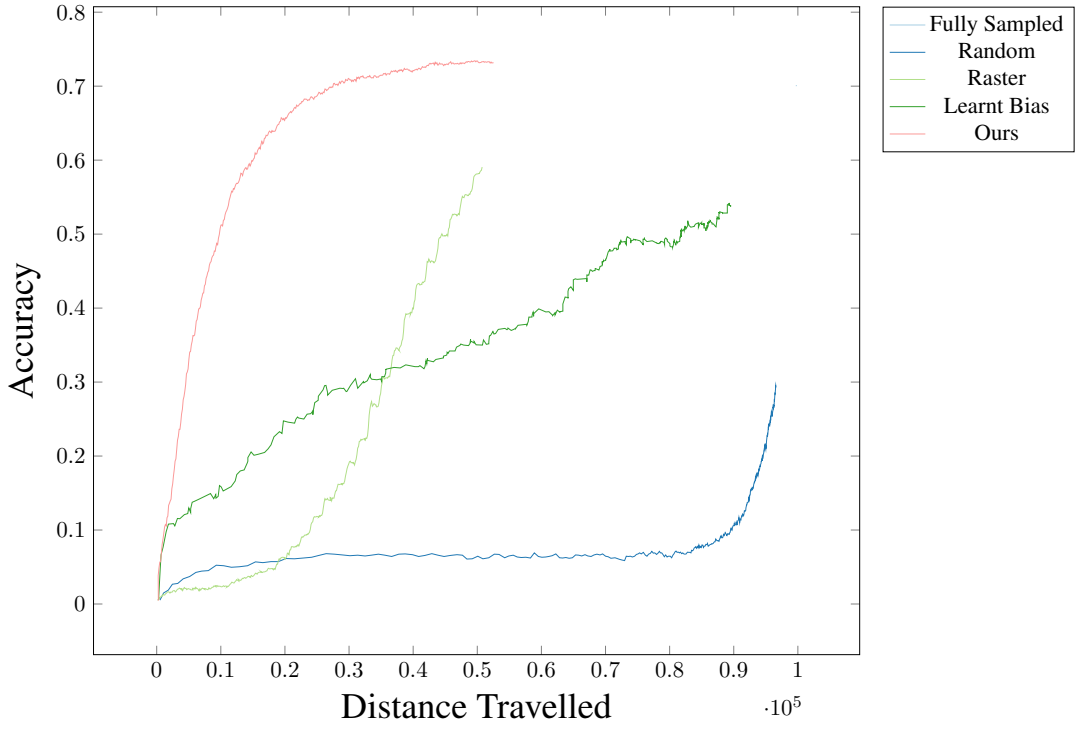


Figure 5.11: Accuracy of classification on CUB-200 dataset compared with scan distance in pixels

5.6 Scan Distance

So far the task metric has been evaluated against the sample rate. The intent here is to minimise the scanning time by minimising the number of samples taken. This is not fully reflective of a real world scenario however, as the scan time will be a function of distance travelled. Therefore a better metric to compare against would be total scan path length.

In previous chapters, as the entire sample map was produced in one go, the scan path would simply be a travelling salesman solution to a given sample map, which is not practically efficient to compute. However, now that each new sample is placed sequentially, the true path length can be recovered, making this a practical comparison.

In fig. 5.11, the scan distance in terms of number of pixels is evaluated. Our method significantly reduces the total length of the scan compared to the other methods. This means that it also a good method for selecting scan patterns in a real-world scenario, where distance between

samples becomes a relevant factor in scan speed. This shows that the samples that it is taking are not only relevant but also relatively close.

5.7 Conclusion

This chapter proposes an algorithm that allows for efficient selection of sample positions for a scanning pixel camera. This allows it to efficiently image scenes for a given computer vision task while still giving comparable performance. This can be adapted to a specific computer vision task, providing better performance than other methods. This allows us to leverage the low-cost, low power advantages of a scanning pixel camera, while reducing the drawback of slow imaging time. Such a camera will therefore become useful in internet of things based systems, where power and data budgets are limited. In the future this can be extended to dynamic scenes and associated tasks such as tracking.

This builds on previous chapters' work and creates a algorithm that allows a scanning pixel camera to image a scene optimally using the minimum amount of samples for a given downstream computer vision task, without any constraints of it having to produce a single image at any given time.

Chapter 6

Conclusions and Future Work

This thesis was designed to exploit the unique advantages of a scanning pixel camera in a computer vision context. These include low-power optics and the ability to focus on a specific area of the scene rather than having to image the entire scene. Inspired by the biological phenomenon of saccades, which are found in mammalian vision systems, this has lead to the development of a scanning pixel camera that is able to only view the relevant parts of an image for a given computer vision task. By only providing relevant information, a solution to the downstream task can be found whilst simultaneously optimising imaging time.

Existing work on scanning pixel cameras in this context is limited as the common operating principle is to use compressive sensing to reconstruct the original image. However, this thesis has eliminated this step completely by directly providing a sampled image that will be useful for the given task. The only similar work to this has been [12], which rather than reconstructing provided an output that could be used for human action recognition. This technique leveraged out-dated machine learning techniques in order to this, and in a modern deep-learning world this would not be sufficient or task generic like the processes presented in this thesis.

Sample positions were represented using a sample map, and the optimisation of this sample map for a given computer vision task was a large focus of this thesis. It was not practical to gather the vast amounts of data data using the new sensor, which would be required for training deep learning models to do this on a broad range of tasks. Instead, a method of treating traditional camera images as if they came from a scanning pixel camera was developed. This allowed us to

leverage a huge amount of already labelled data in order to build scanning pixel camera based computer vision algorithms. Initially, these were developed using the following assumptions:

1. We have existing information on the scene we are sampling.In practice this will not be the case and we will have to dynamically build the best possible sample map based only on pre-existing samples.
2. The scene being imaged is static.However, a scene is likely to move during capture so there is likely to be motion blurring in any acquired scanning pixel camera image.There will also be rolling shutter artefacts however owing to the difference of motion with a traditional camera shutter they would manifest differently.
3. The task can only be performed on a complete frame of samples.As a scanning pixel camera is mimicking the behaviour of saccades, the concept of a frame appears arbitrary and there is no need for such a timing constraint.

However, over the course of the thesis, these were progressively removed to achieve a complete method of producing scan patterns that were optimised to a task. We will now detail how this was done by summarising the conclusions from each chapter in this thesis.

6.1 Establishing Useful Samples

This chapter focussed on finding sampling patterns of scenes that would allow a task to be completed using the minimum amount of samples. This was done with all the assumptions mentioned and provided information that was useful to build the algorithms found in following chapters.

This lead to the establishment of several key points that allowed future chapters to build on this. Namely these are the following,

- That there exists a sampling pattern that requires on average 68% fewer samples in order to complete a task.
- The best way to generate these is to learn them from saliency estimation data, rather than directly from the back-propagation signals provided by the task.

- These sample maps can be generated from a low resolution scan of a scene, giving this technique a method where it can be applied into a real world scenario and still reduce the number of samples taken.

This gave a stable base to adapt these sampling techniques to deal with more scanning pixel camera like data. A key finding was that the best way of generating sample maps was dependent on the task that they were used for. If there was a single clear object that needed to be found within the scene then an external source of saliency based supervision was extremely effective in providing this result. In this case a saliency dataset was used allowing for the final training to focus on adapting the task to learn to deal with the sampled data. However, if a more general view of the scene was required, such as for the instance segmentation task, it was more effective to learn the sampling from the back-propagated signal. This finding was useful in following chapters when developing ways to efficiently train sequential sampling models.

This approach had two major drawbacks to realising a full scanning pixel camera based computer vision system. Firstly, samples are not generated in a predictive manner, which is necessary on a live system. Secondly, we have assumed that the samples are generated a frame at a time, which is an unnecessary restriction given the nature of a scanning pixel camera. Though attempts were made to alleviate the first drawback using a double scan system, a greatly improved method of dealing with these issues was proposed in the two following contributions.

6.2 Where to look next

This chapter addressed some of these issues by expanding the scope of the sampling to dynamic scenes and operated in a predictive manner. To do this sample maps were predicted for each frame in video. A key aspect of this chapter is the network was trained on sequential video however each frame was processed independently, with no gradients flowing between frames. This was done using the ground truth to provide external guidance to each of these frames and effectively kept the computational cost of processing the videos down. The knowledge that it is possible to provide external guidance to train such models was gathered in chapter 3.

This was achieved by using information from the previous frame to infer the sampling for the next frame. This was tested of the video-based tasks of visual object segmentation and human

pose estimation and was able to maintain a 96% relative task accuracy while only using 20% of available samples. This was done by exploiting the information in the task features and using these to simultaneously generate the sample map for the next frame and the task prediction for the current frame.

The end result of this gives a truly predictive system that will work for samples one frame in advance. Furthermore, it works on dynamic data, which is key as the imaging will take time and the data will always be changing. However, this is still required to predict a whole frame's worth of samples at a time, which we deal with in the chapter 5 of this thesis.

6.3 Continuous Sampling

The final contribution of this thesis focused on removing the paradigm of a frame based imaging system. This was a key step in making a working scanning pixel camera based computer vision system as it enabled the sensor to continuously take samples rather than having to produce a frame of samples at a time. This is achieved by having a sample heatmap of likely sampling locations that continuously updates as new samples are taken.

The method for doing this built on that proposed in chapter 4, again using the intermediate feature representation produced by the downstream computer vision task in order to guide the sampling. This was further enhanced by a feature scaling mechanism which encouraged exploration of the scene when sample rates were low and gathered more detail in the scene when sample rates approached a target sample rate.

A key difference is that the system no longer relied on information from previous frames unlike in chapter 4. While this was not necessary to achieve a good result, it may be beneficial as a future improvement to the architecture.

The resulting sampling system had similar performance to the sampling in chapter 3, however this time the sampling was truly predictive. Not only is this close to the original goal of this thesis, but it also allows further refinement on the desired stopping condition, as accuracy can be sacrificed for imaging speed if required. This creates a hardware ready system that would continuously generate new sample positions for a given task. This allows us to leverage the low-cost, low power advantages of a scanning pixel camera, while reducing the drawback of

slow imaging time. Such a camera will therefore become useful in internet of things based systems, where power and data budgets are limited.

6.4 Future Directions for the field

6.4.1 Hardware integration

As this thesis had dealt with a scanning pixel camera in simulation, a clear path forward is integrating the methods that are presented here into a real-world system. This will likely result in dealing with more blurry data as in section 1.3 the assumption was made that the samples were taken without overlap. This may be dealt with by augmenting the training data so that the task network learns to dealt with blurred data without loss of accuracy. Alternatively, the blur could be removed using de-blurring techniques. As the blur is equivalent to a box blur this should be possible with a deconvolution, however in practice this may not be consistent so more advanced techniques such as a Weiner deconvolution [15] may be used.

There is also the question of what to do with the information that can be gathered by sampling while travelling between distant sample positions. Although, this data may be blurry and effectively have low resolution it may sill contain enough effective information to improve the result given by the task and therefore effectively reduce the number of samples required to image the scene for a task. This approach would be contrary to the biological inspiration for a scanning pixel camera, as the information gained by the eye during a saccadic motion appears to be ignored. While the saccades focus the densest area of the eye retina on detailed areas within a scene, the significantly sparser edges of the retina still provide information, particularly for things like sudden motion in the periphery of the field-of-view. Leveraging low resolution information gathered during the motion of the sensor may allow for an effect like this, diverging the scanning pixel camera from its biological inspiration.

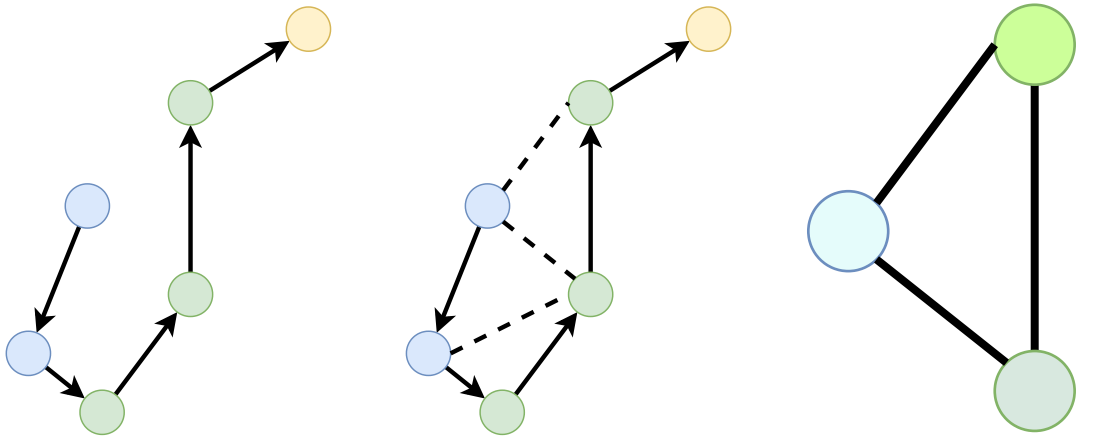
6.4.2 Scanning pixel camera representation

The scanning pixel camera image is accumulated onto a black background, which allows the convolutional network that it is passed into to understand distances between samples. However,

they do not provide useful chromatic information, meaning that the space is not being used efficiently. Furthermore, the choice of a black background may make it difficult to process an image of a dark scene. One way of getting round this is to use some form of partial convolution, such as [38], which would ignore any non-spatial information provided by the background as it would only pass on the convolution result of non-background pixels. While this would not reduce the computation overhead, owing to the parallelised nature of the computation and the need for establishing which pixels have not been scanned, it would reduce the influence that the background would have on the scene.

The problem created by accumulating samples onto a background could be bypassed entirely by operating directly on the sample sequence. This would allow the use of sequence based techniques such as transformer networks, and would significantly reduce the amount of data being processed. This would also make the algorithms resolution agnostic and able to take into account intermediate samples made while travelling to a sample position, thus more effectively representing the data from a scanning pixel camera. However, the major problem here is that the spatial information has been lost. This is detrimental for image processing as the relative locations of pixels are of significant importance. This can be demonstrated through the effectiveness of convolutional neural networks. Convolutional kernels are small relative to the input image allowing them to extract features in at a local level, which can then be pooled to provide a more global understanding of the image.

It is possible to regain this spatial information by using appropriate positional encoding of the samples. However, this may be non-trivial owing to the two dimensional nature of the data. Furthermore, just as a convolutional neural network (CNN) has access to the spatial information at all layers of the network, the positional encoding may have to also be applied to all layers. While this may be easy at first, there may eventually be the requirement to perform a downsampling which would also have to apply to the positional encoding. This, combined with the fact that a task that was traditionally designed for two-dimensional data is now being applied to one-dimensional data, would require novel architectures in order to effectively provide good results.



(a) Orginal graph generated from scan

(b) Graph with addintional semantic links

(c) Downsampled graph for easier processing

Figure 6.1: Graph Neural Network (GNN) based scanning pixel camera data representations

6.4.3 Using Graph Neural Networks

Alternatively, it is possible to consider each sample as a node on a graph and use the edges of the graph to encode the spatial information, as shown in fig. 6.1a. The edge weights can be used to represent an arbitrary distance, rather than being bound to a fixed grid as in the sample map representation found within this thesis. GNNs such as [40] could then be used to perform the downstream tasks. This would allow for nodes to be linked based on semantic content, as in fig. 6.1b, in order to increase the expressiveness of the graph. Further processing would provide a task result based on the graph.

This approach introduces its own challenges namely that there are a large number of nodes that make it computationally expensive to process. One method of combatting this would require the use of graph downsampling in order to reduce this overhead, however, doing this effectively is currently an open research problem.

Utilising graph convolutions to generate sampling predictively, as in chapters 4 and 5, is effectively the addition of nodes to the graph. In its purest form, this means that the representation of scanning pixel camera data becomes a dynamic graph. Graph convolutions on dynamic graphs are currently an area of limited research [67].

6.4.4 A reinforcement learning based approach

Regardless of the representation chosen alternative methods of predicting future samples exist. A major one is using reinforcement learning to predict continuous sampling as this fits nicely into the reinforcement learning problem definition. Under this setup, an agent would be trained to take actions in an environment that would maximise a reward. For a scanning pixel camera, the environment is the scene being imaged, the actions are the motion taken by the sensor and the reward is produced by evaluating performance on the task.

This will particularly apply to chapters 4 and 5 as in both these cases we seek to produce an output based on a sequence of observations of the environment. One of the key challenges for this is that the reward function will have to be simultaneously learnt. This is because the task network that would provide the loss must be trained to correctly process scanned images, so must be trained along with the agent. This may present problems, particularly in early stages of training. Another problem is that each scene imaged is effectively a separate environment, so the agent needs to learn to generalise to many different environments in order to be effective.

6.5 Future Hardware

The scanning pixel camera in this thesis uses a single sensor behind moveable optics. However, there are additional ways to set up this hardware that may provide additional capabilities. For example, it could be combined with a conventional camera in order to provide a global view of the scene that could then be used to help guide the scanning pixel camera to examine regions of interest in further detail. This could be a very low resolution camera, in order to keep the power budget of the overall system down, however, this would still provide useful information.

Also unexamined within this thesis is the possibility of using multiple scanning pixel cameras in an array. The immediate advantage of this is that an entire scene could be imaged much faster. The additional streams of information could be additionally leveraged in order to provide better scan patterns overall while still maintaining many of the advantages of a scanning pixel camera. There are also many computer vision applications that utilise one or more cameras, such as 3D reconstruction. Many of these methods rely on establishing correspondences between pixels

taken from different cameras. Such algorithms would have to be adapted to use a scanning pixel camera as the optical setup is different.

Future hardware implementations could also take further inspiration from the eye, by surrounding a single moving sensor with several static scanning pixel cameras. These would have the effect of providing peripheral vision to the sensor, allowing for better prediction of where to look. This is similar to attaching a scanning pixel camera to a conventional camera, however it further reduces the power requirements and is able to utilise the advantages of using non-imaging optics. One could even go as far as to use dynamic vision sensors as the photodetectors in the peripheral sensors. These would provide information on brightness changes at a high temporal frequency. Combining these with a scanning pixel camera would allow it to respond to rapidly moving objects, further enhancing its capabilities.

Throughout this thesis it has been shown that a scanning pixel camera can effectively be tightly integrated into a computer vision system. With further development of the hardware, as well as the relevant software support, it will be possible to grow the scanning pixel camera into an intelligent sensor that is takes an active role in performing computer vision tasks.

Appendix A

DeepSAUCE Architecture

DeepSAUCE uses EfficientNet-b0 [57] as it's encoder. This gave five layers to the feature pyramid produced by the encoder. The sizes of the feature pyramid are described in table A.1. The encoder was pre-trained on ImageNet. Inputs were normalised using the ImageNet norm prior to being passed in, these are,

$$\mu = [0.485, 0.456, 0.406] \tag{A.1}$$

$$\sigma = [0.229, 0.224, 0.225] \tag{A.2}$$

Each of the feed-forward networks in the transformers had a size of 128.

Encoder feature channels	Total Spatial Reduction	Decoder Transformer heads
16	2	-
24	4	6
40	8	10
112	16	28
320	32	80

Table A.1: Size of the encoder features, and their scale reduction. Also shown is the number for the transformer in the corresponding decoder layer.

Appendix B

Establishing Useful Samples training parameters

Table B.1 shows the training parameters used for different datasets in chapter 3. All models were trained until they converged. All inputs were of size 224×224.

Dataset	# Classes	Learning Rate	Precision	# Epochs	Batch Size
CUB-200	200	0.0001	16	25	75
iNaturalist	10000	0.0001	16	19	50
MSCOCO	91	0.0001	16	33	32
ImageNet	1000	0.0001	16	3	75

Table B.1: Training parameters for different datasets used in each of the experiments in chapter 3.

Appendix C

Additional Downsampling Results

Presented here are results on additional tasks while downsampling the initial input as in section 3.5.

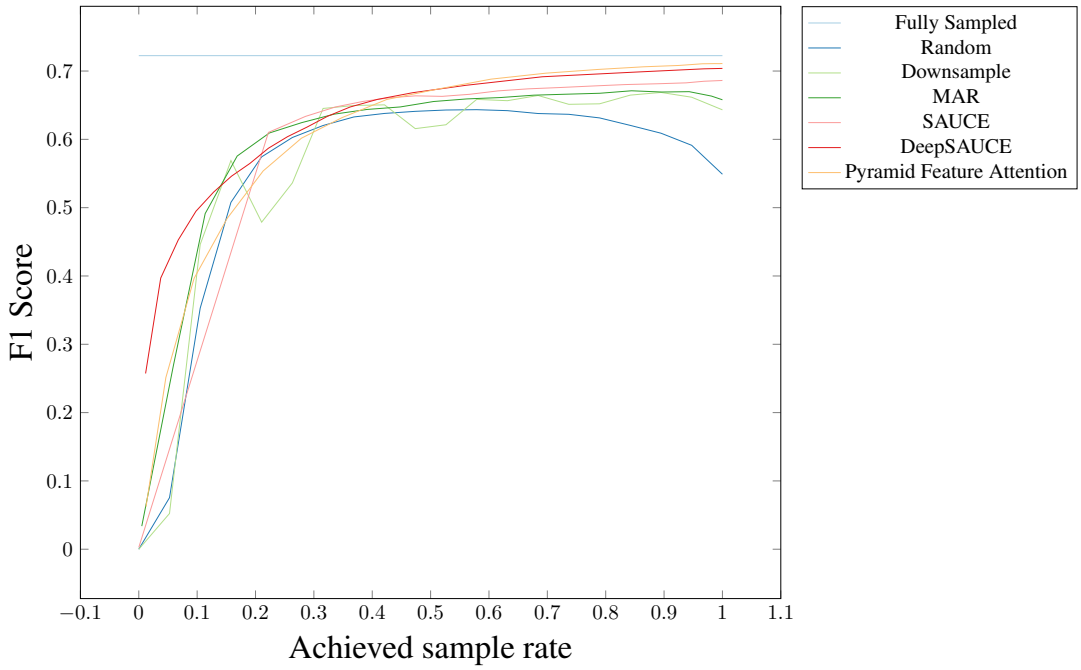


Figure C.1: Evaluation of MSCOCO when the initial view is downsampled by a factor of 4

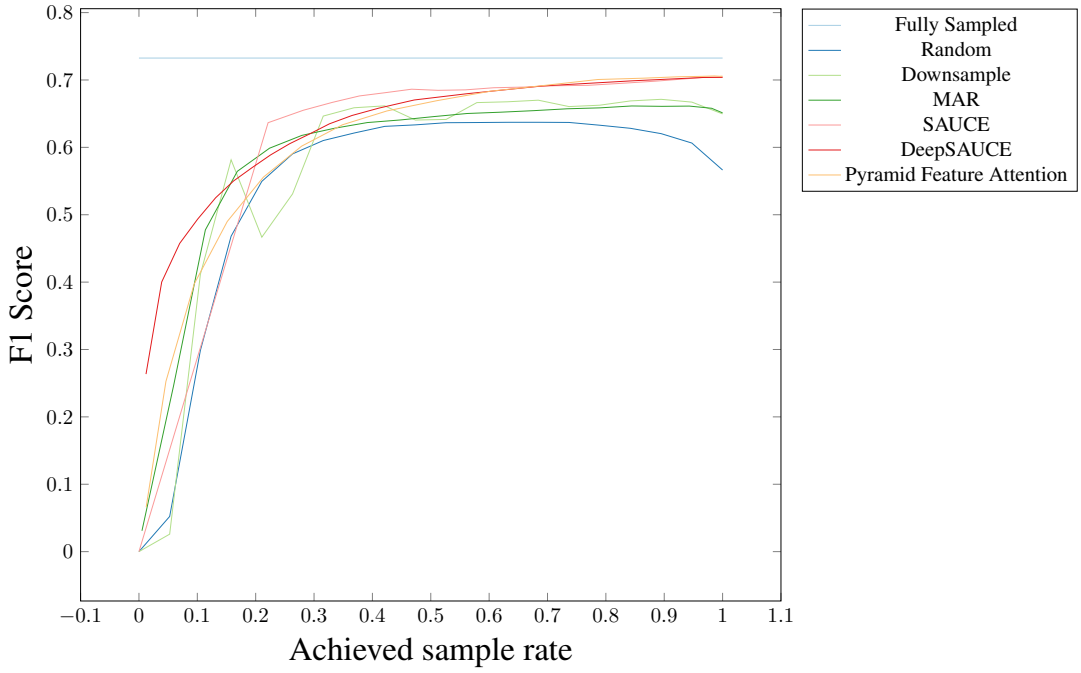


Figure C.2: Evaluation of MSCOCO when the initial view is downsampled by a factor of 10

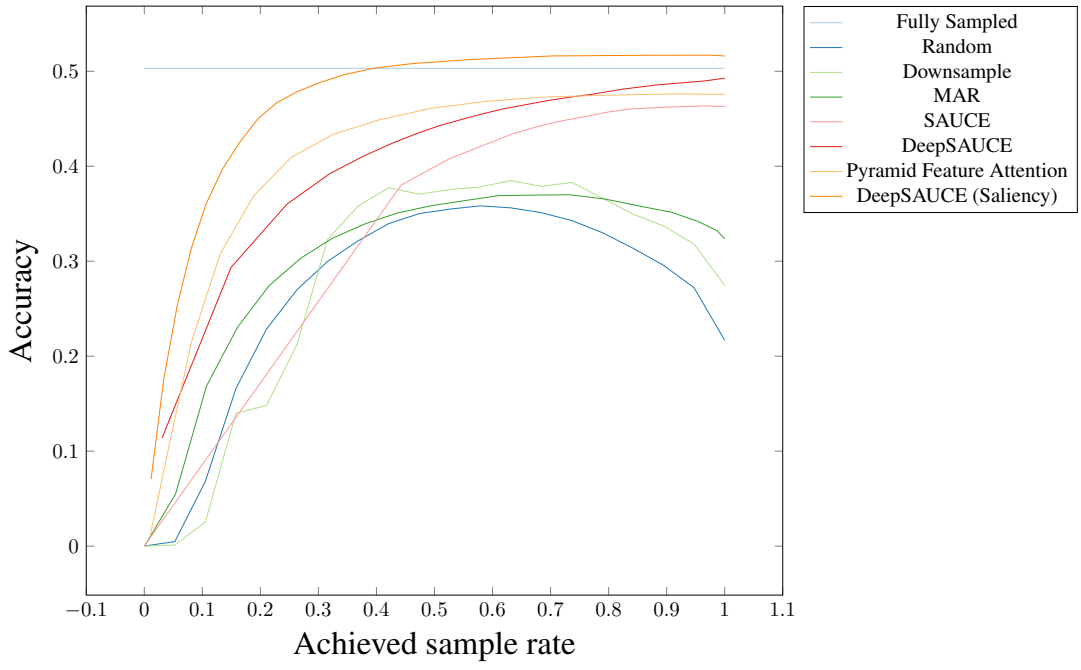


Figure C.3: Evaluation of iNaturalist when the initial view is downsampled by a factor of 4

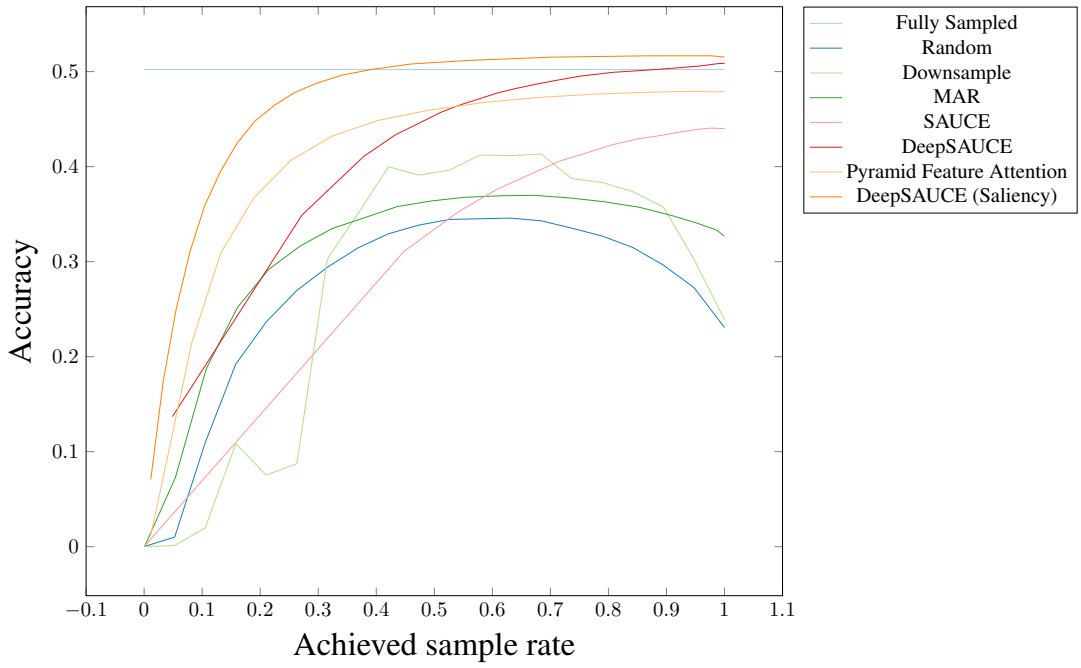


Figure C.4: Evaluation of iNaturalist when the initial view is downsampled by a factor of 10

Appendix D

Where to look next Training parameters

D.1 Task model parameters

In table D.1 we go in to more detail about each of the task models used. Each model has a different sizes of feature pyramid due to differences in the model architecture. For all models the backbone was pretrained on ImageNet [13]. Some models were then additionally pretrained on more task specific datasets. For TransPose features are taken from transformer layers and reshaped to the 3D shape listed.

D.2 Training parameters

We train using the ADAM optimiser [31] with a learning rate of 0.0001, with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. All models were trained for 100 epochs. An exponential decay of 0.98 was applied to the learning rate per epoch. All models were trained using mixed precision [39]. For the tracking task the batch size was 30 and for Joint Prediction is was 20. Videos are generated for each task at a target samplerate of 0.25.

	SSM-VOS	**STM**	**TransPose**
Input size	224×224	224×224	256×192
Backbone	Resnet50 [25]	Resnet50	Resnet-S
Pretrain dataset	DAVIS-17	None	MS-COCO
Feature pyramid	$\begin{bmatrix} 4096 \times 7 \times 7 \\ 2048 \times 14 \times 14 \\ 512 \times 28 \times 28 \\ 256 \times 56 \times 56 \end{bmatrix}$	$\begin{bmatrix} 1024 \times 14 \times 14 \\ 512 \times 28 \times 28 \\ 256 \times 56 \times 56 \end{bmatrix}$	$\begin{bmatrix} 256 \times 32 \times 24 \\ 256 \times 32 \times 24 \\ 256 \times 32 \times 24 \end{bmatrix}$

Table D.1: Additional details for all of the task models used. Feature pyramids given as Channels $\times$ Height $\times$ Width

Bibliography

[1] Earth observing 1 (eo-1), Nov 2000.

[2] Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

[3] Richard G Baraniuk, Thomas Goldstein, Aswin C Sankaranarayanan, Christoph Studer, Ashok Veeraraghavan, and Michael B Wakin. Compressive video sensing: algorithms, architectures, and applications. *IEEE Signal Processing Magazine*, 34(1):52–66, 2017.

[4] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer, 2016.

[5] Emmanuel J Candes, Justin K Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 59(8):1207–1223, 2006.

[6] Hongwei Chen, Zhiliang Weng, Yunhua Liang, Cheng Lei, Fangjian Xing, Minghua Chen, and Shizhong Xie. High speed single-pixel imaging via time domain compressive sampling. In *CLEO: Applications and technology*, pages JTh2A–132. Optical Society of America, 2014.

[7] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[8] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[9] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[10] Robert L Cook. Stochastic sampling in computer graphics. *ACM Transactions on Graphics (TOG)*, 5(1):51–72, 1986.

[11] Mark A. Davenport, Marco F. Duarte, Michael B. Wakin, Jason N. Laska, Dharmpal Takhar, Kevin F. Kelly, and Richard Baraniuk. The smashed filter for compressive classification and target recognition. In *Electronic Imaging*, 2007.

[12] Mark A Davenport, Marco F Duarte, Michael B Wakin, Jason N Laska, Dharmpal Takhar, Kevin F Kelly, and Richard G Baraniuk. The smashed filter for compressive classification and target recognition. In *Computational Imaging V*, volume 6498, pages 142–153. SPIE, 2007.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[14] Atam Prakash Dhawan, Rangaraj M. Rangayyan, and Richard Gordon. Image restoration by wiener deconvolution in limited-view computed tomography. *Appl. Opt.*, 24(23):4013–4020, Dec 1985.

[15] Atam Prakash Dhawan, Rangaraj M Rangayyan, and Richard Gordon. Image restoration by wiener deconvolution in limited-view computed tomography. *Applied optics*, 24(23):4013–4020, 1985.

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly,

Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

[17] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

[18] M. F. Duarte, M. A. Davenport, D. Takhar, J. N. Laska, T. Sun, K. F. Kelly, and R. G. Baraniuk. Single-pixel imaging via compressive sampling. *IEEE Signal Processing Magazine*, 25(2):83–91, 2008.

[19] Gamaleldin F. Elsayed, Simon Kornblith, and Quoc V. Le. Saccader: Improving accuracy of hard attention models for vision, 2019.

[20] Tongle Fan, Guanglei Wang, Yan Li, and Hongrui Wang. Ma-net: A multi-scale attention network for liver and tumor segmentation. *IEEE Access*, 8:179656–179665, 2020.

[21] Guillermo Gallego, Tobi Delbrück, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew J Davison, Jörg Conradt, Kostas Daniilidis, et al. Event-based vision: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(1):154–180, 2020.

[22] Jacob Gildenblat. Jacobgil/pytorch-grad-cam: Advanced ai explainability for computer vision. support for cnns, vision transformers, classification, object detection, segmentation, image similarity and more., May 2017.

[23] Amirhossein Habibian, Davide Abati, Taco S Cohen, and Babak Ehteshami Bejnordi. Skip-convolutions for efficient video processing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2695–2704, 2021.

[24] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16000–16009, June 2022.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[26] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[27] Catherine F. Higham, Roderick Murray-Smith, Miles J. Padgett, and Matthew P. Edgar. Deep learning for real-time single-pixel video. *Scientific Reports*, 8(1):2369, Feb 2018.

[28] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset, 2018.

[29] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[30] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black. Towards understanding action recognition. In *International Conf. on Computer Vision (ICCV)*, pages 3192–3199, December 2013.

[31] Diederik P Kingma and Jimmy Lei Ba. Adam: A method for stochastic gradient descent. In *ICLR: International Conference on Learning Representations*, 2015.

[32] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[33] Kuldeep Kulkarni and Pavan Turaga. Reconstruction-free action inference from compressive imagers. *IEEE transactions on pattern analysis and machine intelligence*, 38(4):772–784, 2015.

[34] Matthias Kümmerer, Thomas S. A. Wallis, and Matthias Bethge. Saliency benchmarking made easy: Separating models, maps and metrics. In Vittorio Ferrari, Martial Hebert,

Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, Lecture Notes in Computer Science, pages 798–814. Springer International Publishing.

[35] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128$\times$ 128 120 db 15 μs latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits*, 43(2):566–576, 2008.

[36] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. mid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[37] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.

[38] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *Proceedings of the European conference on computer vision (ECCV)*, pages 85–100, 2018.

[39] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

[40] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5115–5124, 2017.

[41] Anh Nguyen, Thanh-Toan Do, Darwin G Caldwell, and Nikos G Tsagarakis. Real-time pose estimation for event cameras with stacked spatial lstm networks. *arXiv preprint arXiv:1708.09011*, 3, 2017.

[42] Xiaoqi Nong and Simon Hadfield. Asl-slam: An asynchronous formulation of lines for slam with event sensors. In *2022 The 9th International Conference on Industrial Engineering and Applications (Europe)*, pages 84–91, 2022.

[43] Peter O'Connor and Max Welling. Sigma delta quantized networks. *arXiv preprint arXiv:1611.02024*, 2016.

[44] Seoung Wug Oh, Joon-Young Lee, Ning Xu, and Seon Joo Kim. Video object segmentation using space-time memory networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[45] George Papandreou, Tyler Zhu, Liang-Chieh Chen, Spyros Gidaris, Jonathan Tompson, and Kevin Murphy. Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model. In *Proceedings of the European conference on computer vision (ECCV)*, pages 269–286, 2018.

[46] Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. In *EMNLP*, 2016.

[47] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Computer Vision and Pattern Recognition*, 2016.

[48] Harish Guruprasad Ramaswamy et al. Ablation-cam: Visual explanations for deep convolutional network via gradient-free localization. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 983–991, 2020.

[49] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[50] N. Sayiner, H. V. Sorensen, and T. R. Viswanathan. A level-crossing sampling scheme for a/d conversion. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 43(4):335–339, 1996.

[51] Cedric Scheerlinck, Henri Rebecq, Daniel Gehrig, Nick Barnes, Robert Mahony, and Davide Scaramuzza. Fast image reconstruction with an event camera. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, March 2020.

[52] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[53] C. E. Shannon. Communication in the Presence of Noise. *Proceedings of the IRE*, 37(1):10–21, jan 1949.

[54] Saurabh Singh and Shankar Krishnan. Filter Response Normalization Layer: Eliminating Batch Dependence in the Training of Deep Neural Networks, 2019.

[55] J Springenberg, Alexey Dosovitskiy, Thomas Brox, and M Riedmiller. Striving for simplicity: The all convolutional net. In *ICLR (workshop track)*, 2015.

[56] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5693–5703, 2019.

[57] Mingxing Tan and Quoc Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *International Conference on Machine Learning*, pages 6105–6114, 2019.

[58] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1653–1660, 2014.

[59] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[60] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *NIPS*, 2017.

[61] Sagar Vaze, Kai Han, Andrea Vedaldi, and Andrew Zisserman. Open-set recognition: A good closed-set classifier is all you need. In *International Conference on Learning Representations*, 2022.

[62] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[63] Brian Wandell and Stephen Thomas. Foundations of vision. *Psyccritiques*, 42(7), 1997.

[64] Lijun Wang, Huchuan Lu, Yifan Wang, Mengyang Feng, Dong Wang, Baocai Yin, and Xiang Ruan. Learning to detect salient objects with image-level supervision. In *CVPR*, 2017.

[65] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 4724–4732, 2016.

[66] Wikipedia. File:Nonimaging Optics-CPC Acceptance.png — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=File%3ANonimaging%20Optics-CPC%20Acceptance.png&oldid=468517395`, 2022. [Online; accessed 18-May-2022].

[67] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021.

[68] Ning Xu, Linjie Yang, Yuchen Fan, Dingcheng Yue, Yuchen Liang, Jianchao Yang, and Thomas Huang. Youtube-vos: A large-scale video object segmentation benchmark. *arXiv preprint arXiv:1809.03327*, 2018.

[69] Jun Yang, Wei EI Sha, Hongyang Chao, and Zhu Jin. High-quality image restoration from partial mixed adaptive-random measurements. *Multimedia Tools and Applications*, 75(11):6189–6205, 2016.

[70] Sen Yang, Zhibin Quan, Mu Nie, and Wankou Yang. Transpose: Keypoint localization via transformer. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[71] Yi Yang and Deva Ramanan. Articulated human detection with flexible mixtures of parts. *IEEE transactions on pattern analysis and machine intelligence*, 35(12):2878–2890, 2012.

[72] Xiaoyu Yue, Shuyang Sun, Zhanghui Kuang, Meng Wei, Philip HS Torr, Wayne Zhang, and Dahua Lin. Vision transformer with progressive sampling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 387–396, 2021.

[73] Dario Zanca and Marco Gori. Variational laws of visual attention for dynamic scenes. *Advances in Neural Information Processing Systems*, 30, 2017.

[74] Yan Zhang, GM Godaliyadda, Nicola Ferrier, Emine B Gulsoy, Charles A Bouman, and Charudatta Phatak. Slads-net: Supervised learning approach for dynamic sampling using deep neural networks. *Electronic Imaging*, 2018(15):131–1, 2018.

[75] Ting Zhao and Xiangqian Wu. Pyramid feature attention network for saliency detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[76] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13001–13008, 2020.

[77] Wencheng Zhu, Jiahao Li, Jiwen Lu, and Jie Zhou. Separable structure modeling for semi-supervised video object segmentation. *IEEE Transactions on Circuits and Systems for Video Technology*, 2021.