MASTER'S THESIS 2026

Classification of Small Thermal Images with Transformers

Elin Persson

ISSN 1650-2884
LU-CS-EX: 2026-04

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2026-04

Classification of Small Thermal Images with Transformers

Klassificering av Små Termiska Bilder med Transformers

Elin Persson

Classification of Small Thermal Images with Transformers

(An Evaluation for Security Applications)

Elin Persson
el8065pe-s@student.lu.se

February 26, 2026

Master's thesis work carried out at Axis.

Supervisors: Thomas Ekdahl, thomas.ekdahl@axis.com
Pierre Nugues, pierre.nugues@cs.lth.se

Examiner: Eren Aksoy, eren.aksoy@cs.lth.se

Abstract

As security systems become increasingly sophisticated, accurate human detection is critical to prevent costly false alarms. Thermal cameras, which produce low-resolution grayscale images through white-hot mapping, make it challenging to distinguish humans from large animals. Axis currently faces this issue, as their existing classifier frequently misidentifies humans, triggering unnecessary alerts. This thesis investigates transformer-based architectures as an alternative to improve classification accuracy on small thermal images. We compare several transformer models, benchmark them against convolutional neural networks (CNNs), and explore how different architecture designs affect performance. Our study highlights the trade-off between model size and accuracy, with a focus on solutions that are lightweight enough for embedded systems. The results identify a model that reduces false alarms while maintaining computational efficiency, offering a practical improvement for real-world security systems. This work contributes to the understanding of human detection in thermal imagery and provides guidance for designing efficient, high-accuracy classifiers in constrained environments.

Keywords: MSc, Image classification, thermal, low resolution images, vision transformers, CNN, security, false alarm reduction

Acknowledgements

I would like to thank Nicolas Audoux and Natacha Ruchaud for all their support during this project. You have taught me a lot, and I have really enjoyed working with you. I would also like to thank my supervisors at Axis, Laurent Giulieri and Thomas Ekdahl, for making it possible for me to do this thesis partly in France. Thanks also go to my supervisor at LTH, Pierre Nugues, for his valuable advice throughout the project and for taking the time to answer all my questions. Finally, I would like to thank my examiner, Eren Aksoy, for the constructive feedback on this thesis.

Contents

Chapter 1
Introduction

1.1 Introduction

Security systems are rapidly advancing and becoming more and more automated. Input devices include optical cameras, thermal cameras, radar, motion sensors and laser mapping. Object detection has become an essential part of these security systems to be able to send alarms when specific objects, such as humans, are detected. In this work, we will focus on the classification of the detected objects in thermal images.

Optical cameras are more commonly used than thermal cameras, but the latter have a different purpose. Thermal cameras distinguish objects by their relative temperatures within the image. Therefore, they make it easier to detect humans and animals, which generally radiate more heat than their surroundings, at least in Sweden, where the climate is mild. This is because they allow humans and other objects that radiate heat to be seen in complete darkness and sometimes even through obstacles like trees and bushes. This is something that optical cameras cannot do. Thermal cameras are also more robust in that they are invariant to shadows, color changes, and glare. When choosing between optical cameras and thermal cameras, it all comes down to what you are interested in capturing, and sometimes, both variants are used together.

While optical cameras usually have three color channels, red, green, and blue (RGB), thermal cameras have only one. Instead of relying on visual light, thermal cameras capture the infrared energy emitted by objects that are present in the image (Nguyen et al., 2021). The temperature distribution captured within the scene can then be mapped to a color palette (Agrawal and Karar, 2018). This corresponds to false colors as opposed to true colors (RGB) from optical cameras. The thermal images in our dataset are white-hot, rendering single-channel images in grayscale, where whiteness increases with temperature.

The two most efficient architectures to classify objects in images are convolutional neural networks (CNN) and vision transformers (ViT) and they have a history of competition in image classification. Transformer networks have famously achieved state-of-the-art perfor-

mance on natural language processing (NLP) tasks owing largely to the powerful attention mechanisms that are a key part of their structure. This architecture proved very efficient in image classification tasks as well (Dosovitskiy et al., 2021).

Axis is currently using a CNN-based classifier within their thermal cameras, with the purpose of classifying already detected objects. However, the classifier sometimes confuses humans with large animals or vehicles. Figures 1.1 through 1.4 show an example of this, taken from one of Axis' own cameras.

Transformers are competitive or better than CNNs on large-scale RGB benchmarks (Maurício et al., 2023). Nonetheless, their effectiveness on small, low-texture thermal images remains unclear. Thermal cameras naturally have a lower resolution than RGB cameras, and the images sent to the classifier are, in our case, very small. This is because Axis perform motion-based object detection prior to classification. Therefore the images are limited to the size of the bounding box of the detected object. Figure 2.3 shows what they look like. Additionally, when these images are artificially resized, they get warped and noisy.

In this work, we investigate the performance of different transformer models on small thermal images, and compare them with a baseline CNN. Furthermore, the comparison will take into account not only the achieved overall accuracy, but also the model size in number of parameters, the recall and precision scores of the person class as well as the inference time, estimated in number of floating point operations (FLOPs). This is because the main objective of this classifier is to always detect people, whilst still keeping the number of false alarms to a minimum. The chosen model also needs to have potential to run on an embedded system.

The key findings of this work are that transformers can perform better than CNNs on classification tasks with small thermal images and that FLOPs and number of parameters in transformers can be reduced whilst keeping results competitive. This shows that transformers have as much potential to run on embedded systems as CNNs, despite transformers being known for computational heaviness. Above all, the main conclusion from this report is that MaxViT performed the best for our use case, increasing overall accuracy with 1.16 percentage points compared with Axis' original classifier. Additionally, we saw that MaxViT also decreased false alarms without an increase in missed person detections, because both recall and precision of the person class improved.

The methodology used for this analysis consists of three parts. First, we choose the transformers that we think are interesting for our use case, and finetune the pretrained versions of them on Axis' large dataset. Second, we choose our favorite models based on the overall accuracy as well as the recall and precision of the person class, and we experiment with their hyperparameters and architectures using Optuna. We try different training approaches, like training from scratch on ImageNet-1k and training with distillation. Finally, we evaluate our models on another dataset to assess generalization.

Our contributions are threefold: (1) we advance knowledge of image classification in the one-channel, low-resolution domain; (2) we investigate how transformer architectures can be adapted to resource-constrained embedded systems through architecture optimization; and (3) we provide an empirical evaluation of transformer models on small thermal images.

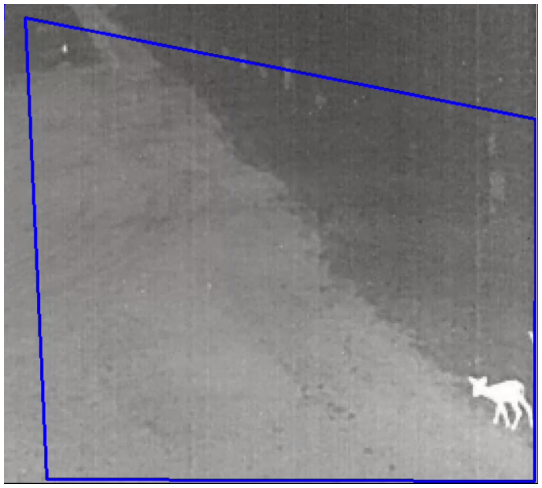


Figure 1.1: The image shows a deer through the lens of one of Axis' thermal cameras. It is yet to be detected. The blue bounding box shows the area where we are able to detect objects.

Figure 1.2: Here the initial deer has been detected but not yet classified, as indicated by its yellow bounding box. It has also been joined by two deer friends.

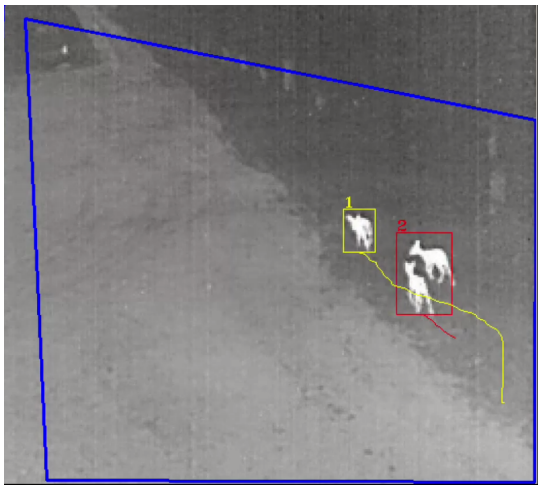


Figure 1.3: The two deer to the lower right get detected as one object and classified as a human, as indicated by their red bounding box. This is an example of when Axis' original CNN classifier confuses animals with humans, triggering a false alarm.

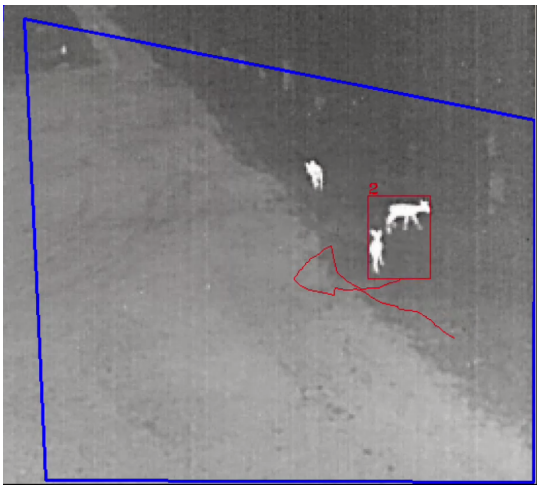


Figure 1.4: The classifier's confusion continues, and the detection of the first deer is lost.

1.2 Research questions

The goal of this thesis is to answer the following questions:

- Can a transformer model outperform Axis' original classifier while scaled down to similar size?

- What training method is the most successful for classification of small thermal images, finetuning or distillation?
- Is a transformer-based architecture a better choice than a CNN for classification of small thermal images under size constraints?

Chapter 2
Datasets

Open source datasets of thermal images are much less common than those of RGB images, and can be very hard to find (Qazi et al., 2025). Especially when you are interested in a certain type of color mapping and certain objects to classify, like we are. A reason for this might be that thermal image data is more difficult to produce, since it requires having a thermal camera. Nevertheless, there are tricks to create an artificial thermal dataset. For instance by using generative AIs like GANs to synthesize thermal images (Ulusoy et al., 2022).

However, applying a grayscale transformation to RGB images will not make the images look equivalent to white-hot thermal images. For RGB images, a grayscale transformation will depend on the brightness of the image. The brightest parts of the image will be the whitest, and the parts of the image with less light intensity will be darker gray. It is clear that this kind of grayscale transformation of RGB images does not produce images equal to that of white-hot thermal images, where the color mapping is based on the relative temperature within the image. The white-hot color mapping ensures that the warmest part of the image will always be the whitest, and then the color gets darker with the decrease of temperature. This means that an animal photographed outside, in cold climate, will look very different on a thermal image versus a grayscale RGB image. Despite this, we see potential benefits of training on entirely grayscale data, like being able to reduce the number of input channels.

In this report, we will work with three datasets. Our two main datasets are confidential because they are the property of Axis, but one dataset is completely public. We describe them in the sections below.

2.1 Axis proprietary datasets

2.1.1 Axis' large dataset

Axis' large dataset consists of labeled data with image sizes ranging from size 9×29 and up. Their training and test data consists of a mixture of public datasets and proprietary data. In

this report, this dataset will be referred to as "Axis' large dataset" and the train- and test splits will be referred to as "Axis' large train set" and "Axis' large test set" respectively. The data is a mixture of thermal and optical images. See Figure 2.1 for an example of a white-hot thermal image. This dataset has around 500,000 images and contains five classes labeled from 0 to 4: animals/false alarm, bikes, people, vehicles and crawling people. About 480,000 images are used for training and the rest for testing. The dataset is not balanced between the classes.

Figure 2.1: An example of a white-hot thermal image. Notice how the cat's nose is the coldest part of the image, aside from the background.

2.1.2 Axis' small dataset

For the final evaluation of all our classifier models, we will use a dataset that mainly consists of large animals, such as rhinos, deer and elephants. Some vehicles are also present in this dataset, but the number of samples of that class is significantly smaller than the number of animal samples. Large animals are currently the largest cause of false alarms for Axis' original classifier, because they are easily confused with humans in thermal images.

This dataset is made by Axis and it is also confidential. It consists of about 80,000 images, where around 50,000 are used for training and the rest for testing. The images are both thermal and RGB. This dataset will be referred to as "Axis' small dataset" in this report, and the train and test splits will be referred to as "Axis' small train set" and "Axis' small test set" respectively. In Figure 2.2 is an example of what the images in this dataset looks like. Using the same class mapping as previously, this dataset solely contains images of class 0 (animals/false alarm) and class 3 (vehicles).

Data collection

When generating data, Axis first leverage their motion-based object detection to extract objects by cropping the image around the bounding box that is generated around detected object. These cropped images, which are very small and irregular in size, populate the datasets that Axis have made themselves using footage from Axis cameras. See Figure 2.3 for some examples of these cropped, bounding box images. See Figure 2.4, where the ambiguity of

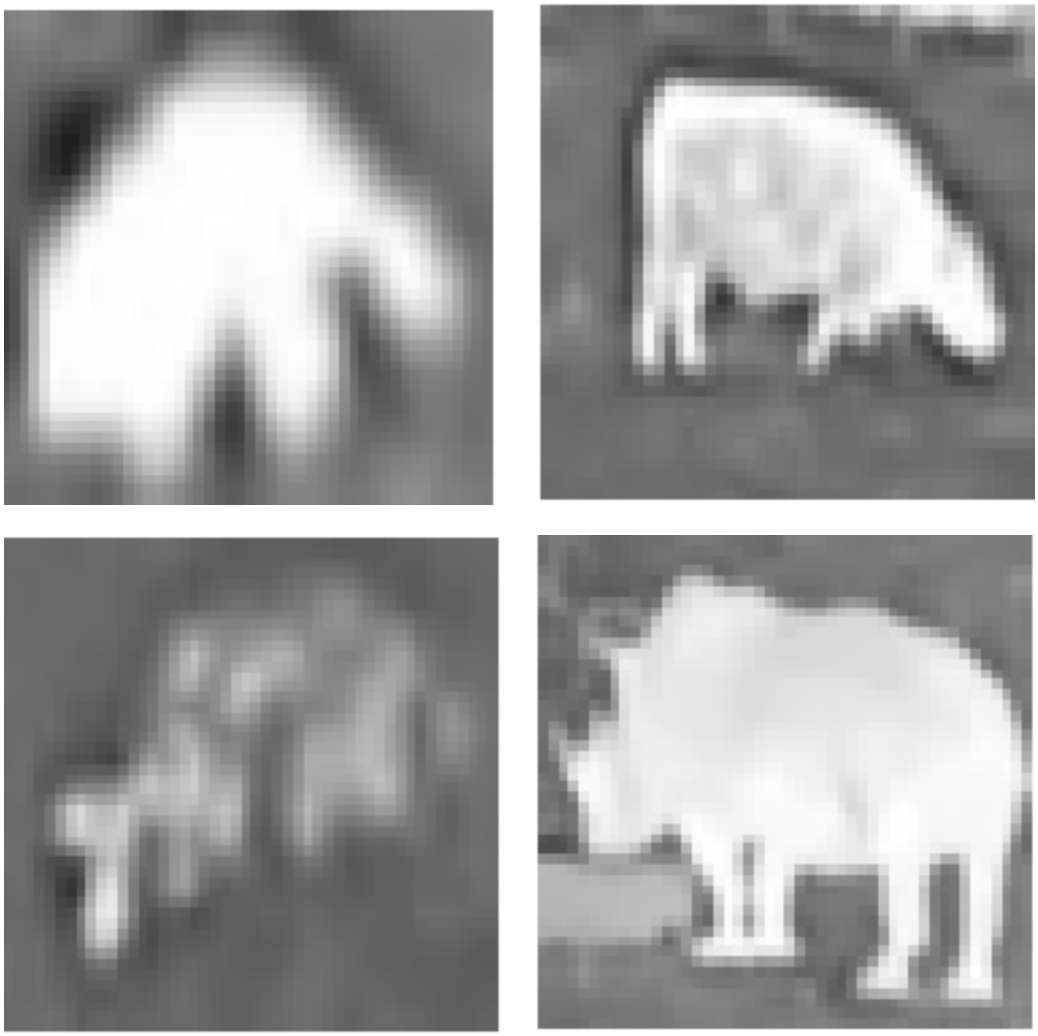


Figure 2.2: Here are four images from Axis' small dataset of animals, the images are all label 0. Top left depicts a cat, top right is a cow, bottom left shows a fox and bottom right is a rhinoceros.

thermal images is further illustrated, showing how easily a large animal can be mistaken for a human.

Image standardization

In addition, Axis complemented proprietary data by adding selected public image databases. All images are then resized to a standardized size, which is quite small. For some models, we are able to use this standardized size, but for others, we must artificially scale up all images to match the models' input sizes, resulting in further loss of resolution. All transformer models that we do not train from scratch in this report have been pretrained on images larger than the ones in our data. The datasets contain both thermal images and regular RGB images. When collecting data, a usual setup is to have two cameras, one regular and one thermal, set up to have the same view. That way, the collected images have two occurrences (one in color and one thermal) adding more variation to the dataset as well as increasing its size with

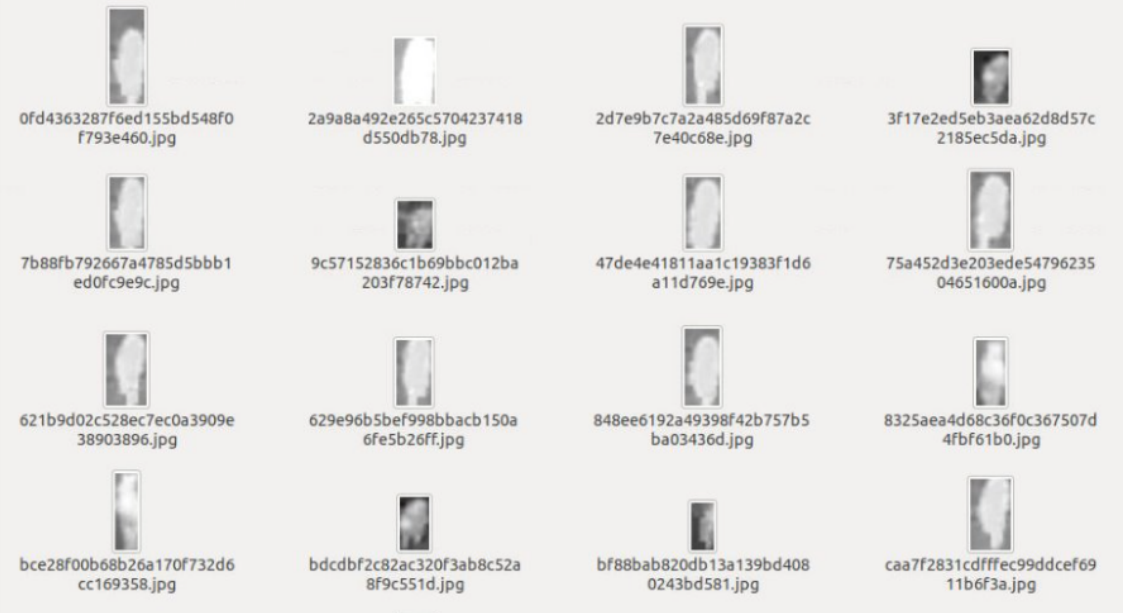


Figure 2.3: Here are some examples of what the input images that are sent to Axis' classifier look like, before they are resized. Each image is only the size of the bounding box of the object.

minimal effort.

2.2 The FLIR dataset

Any further details of Axis' large dataset is confidential. Therefore, we have also performed the initial experiments on a public dataset of thermal images (Imaging, 2024). This dataset is from FLIR[1] and it contains data of dogs, humans and vehicles. When using this dataset, we first resize it to a smaller size, so that the images are comparable with the ones in Axis' large dataset. Then, we crop out all objects by their bounding box and perform new annotations. This is to mimic Axis' own data where each image normally contains only one object to be classified. Additionally, this approach increases our dataset slightly, which is good. In Figure 2.5, some images from our cropped version of the FLIR public dataset are displayed.

Even though the FLIR dataset only has three classes, we use the same class mappings as for Axis' large dataset. This means that the crawling person class and the bike class will both be vacant in the public experiments. However, our main point of interest is for the classifier to be able to distinguish humans from animals and vehicles, and these classes are all present in the FLIR dataset. Therefore it will suffice as a public benchmark.

Of course, there are other differences that are important to consider. Primarily, the FLIR dataset is much smaller and less varied than Axis'. Uncropped, it has a total of 9391 images. Initially, it is divided into a training set, a validation set and a test set of 7083, 2129 and 179 images respectively. We chose to combine the validation and test sets into one, larger test set. The reason for this is that Axis' datasets also are divided that way. Additionally, we got a slightly more proportional test set compared with the training set for the FLIR dataset. It

[1] `https://universe.roboflow.com/thermal-imaging-0hwfw/flir-data-set`

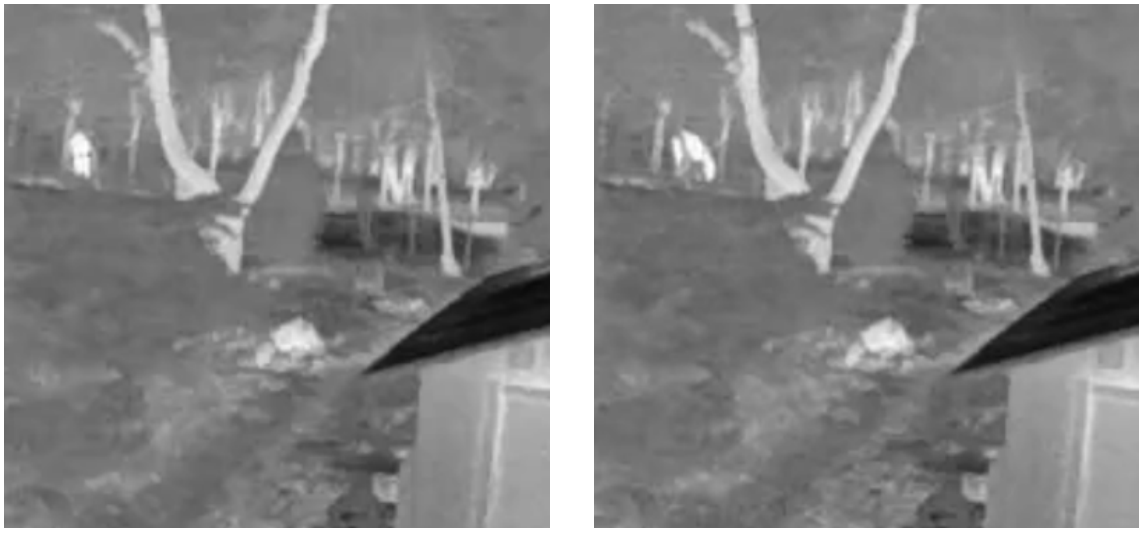


Figure 2.4: In this thermal image (left image) from one of Axis' own cameras, it appears as if there is a person standing in the left side of the background, in between some trees. However, in the next instance, the alleged "person" turns and we can see that it is actually not a person, but a horse (right image). This displays the ambiguity of thermal images and the difficulties that can arise for classification of humans versus large animals.

Table 2.1: FLIR Dataset Split – 3 Classes

Class	Description	Train	Test
0	Animal / False Alarm	356	72
2	Person	45179	5811
3	Vehicle	45272	9525
Total		90807	15408

is important to consider that for the public experiments the amount of data will skew the results, making the models prone to overfitting.

Secondly, the dataset from FLIR is unbalanced. It is biased towards the person class since it contains significantly more images of that class than the other two classes. After cropping out all objects from the dataset, the final distribution of the test and train splits are presented in table 2.1. The same class mappings as used with Axis' large dataset are leveraged. For this reason the animal/false alarm class is labeled 0, the person class is labeled 2 and the vehicles get the class label 3.

Lastly the FLIR dataset only contains thermal images and no optical ones. In Axis' large dataset optical images are leveraged to introduce more variety into the model, but the public dataset consists entirely of one-channel, white-hot images. This leads us to believe that the models trained on the public dataset will have much worse generalization ability. Even so, that might go unnoticed since the test set has the same properties as the training set.

Figure 2.5: Here are some example images from the FLIR public dataset after having made smaller crops of all the objects. The top- and bottom left images have the label 3, vehicle, and the top- and bottom right images have the label 2, person. As evident in the bottom right image, sometimes multiple objects end up in the same image anyway, although it is rare. The two objects are unfortunately still classified as one object when this occurs. In this case we have two objects of the same class, so it will not be a devastating issue.

Chapter 3
Previous work

3.1 Knowledge gap

There are very few publications that investigate thermal image classification using transformer-based architectures. The most closely related work is the study by Nunnari and Calvari (2025), which compares Vision Transformers and convolutional neural networks for the classification of volcanic activity in thermal imagery. The authors conclude that transformers can offer modest accuracy improvements over CNNs, for example reporting slightly higher performance for Swin compared to their baseline CNN models. However, their CNN architectures (e.g., AlexNet and ShuffleNet) exhibit significantly higher computational efficiency.

A key limitation of this study is that it only considers a highly specific classification task, since all classes correspond to different volcanic phenomena. It is not at all related to what is investigated in this thesis, namely the classification of more common everyday objects (humans, animals, vehicles and bikes).

Furthermore, the thermal images used by Nunnari and Calvari are pseudo-colored (rainbow), meaning that the underlying single-channel thermal signal is mapped to three artificial color channels. Such representations introduce camera-dependent color mappings and artificial correlations that are not physically meaningful, and may therefore bias model learning and evaluation. In contrast, white-hot thermal images preserve a direct and monotonic relationship between pixel intensity and temperature. Meaning that even though both this thesis and Nunnari and Calvari's work use thermal images, the domains are fundamentally different.

Finally, existing works that investigate thermal image classification does not explore architectural scaling or optimization when comparing CNNs and transformers. There are however RGB classification works that do this, but for thermal image classification, this is missing. Transformer models are generally larger and more computationally demanding than CNNs. As a result, conclusions regarding model efficiency or superiority are confounded by scale rather than architecture.

This thesis addresses these gaps by studying the classification of semantically diverse real-world objects (humans, animals, bikes and vehicles) in white-hot thermal imagery, while explicitly optimizing transformer architectures to comparable model sizes. This enables a more methodologically fair comparison between CNNs and transformers in the thermal domain.

3.2 Classification of thermal images

Image classification is a core problem in computer vision, historically dominated by convolutional neural networks (CNNs) because of their strong inductive biases that effectively extract local spatial features. More recently, transformer-based architectures, which process images as sequences of patches and use attention to model long-range dependencies, have emerged as a competitive alternative in classification tasks. Currently, these two architectures are the most promising for image classification tasks. In the following sections, we will review different versions of these architectures.

Classification in thermal images differs a bit from classification in RGB images. It poses certain challenges that are otherwise absent. One of these challenges relates to the color mapping (Wang et al., 2010). Thermal data is usually represented as a single grayscale channel that reflects temperature rather than visible color and generally, the resolution is lower than in RGB images. This can make feature learning more difficult, since that is dependent on identifying key details of objects in order to classify them. Another challenge is the limited availability of large annotated thermal datasets, which makes it harder to train larger, more complex models, which require more data. These differences mean that results from RGB image classification do not always transfer directly to the thermal domain, and to understand how modern architectures perform on small thermal images further evaluation is needed.

3.3 Convolutional neural networks

Axis' baseline model has a convolutional neural network (CNN) architecture. Specifically it has a residual network (ResNet) with a number of convolution blocks each containing a number of convolutions. The number of parameters will remain undisclosed, but it is of smaller or equal range to the transformer models we test. With respect to Axis, any more intimate details of their current classifier will not be shared in this report. Instead, a public ResNet model will be used as an alternative benchmark. We will also include EfficientNet in this study because of its competitive performance and compact size.

CNNs contain three different types of layers, convolutional-, pooling- and fully connected layers O'Shea and Nash, 2015. The first layer of the CNN is the input layer, and it holds the pixel values of the input image. In the convolutional layer, a learnable kernel is used to process the input. A kernel is like a small, local receptive filter that strides over the input image. At each position, the kernel performs a linear operation, often a dot product between its weights and the input values of the local image patch. In a CNN, there may be several kernel filters focused on identifying different features. It is the weights of the filter that determines what kind of features it can pick up on. After the dot products have been constructed, the bias term is added. Each computation that takes an input value and maps it to an output value, is commonly referred to as a neuron. The bias term regulates how easily a

neuron is activated. For instance, if the bias term is a positive value, it will make the neuron activate even for values close to zero, and if it is negative it will make the neuron harder to activate. The bias is a learnable scalar vector and each kernel filter has its own bias term that gets added to every calculated dot product. That makes up the raw feature map. At the end, a non-linear activation function, like rectified linear (ReLu) or sigmoid is applied to the raw feature map. It depends a bit on the activation function, but generally, if the output of this function is large for a neuron, it signifies that the neuron is activated. If the output is very small or close to zero, it means that the neuron is inactive or even dead. The activation of a neuron is a representation of whether the neuron found the pattern that it was specifically trained to look for. The output is the final feature map which is in the shape of a multidimensional tensor.

In the pooling layer, a feature map's spatial dimension is reduced. Pooling concatenates the information in a local part of the image. Pooling works with a pooling window that strides over the feature map. At every position, all the values within the window are summarized to a single value, for instance by taking the max or average value. This reduces the spatial dimension of the feature map while keeping the channels untouched. Pooling helps prevent overfitting by simplifying activations and reducing complexity as the network gets deeper.

In a fully connected layer, also called a linear layer, all input units are linearly mapped to all output units. It takes a multidimensional tensor as input and flattens it into a vector. Flattening is a reshaping operation that converts a multi-dimensional tensor into a one-dimensional vector by combining all its elements. Generally for image classification you will have a 3D tensor with channels, height and width after a convolutional layer. Flattening then produces a single vector with the length of channels times height times width. In a CNN, one or more fully connected layers are commonly used in the classification head, *i.e* the output layer that has the purpose of outputting class logits/probabilities for the final prediction. In this layer, all local information from all different kernel patches are combined to construct a global understanding of the image. The very last layer in the CNN is a fully connected layer where the output dimension is equal to the number of classes. If we have ten classes, then for every input the fully connected layer will produce ten logits, which act as a score for each class. These scores are then sent into a softmax function that converts them to class probabilities for each input. Just as expected, all of these class probabilities will sum to 1 after the softmax function. Generally a high logit will convert to a high probability for that class. A given input will be classified as whatever class that got the highest probability by the softmax function.

3.3.1 ResNet-10

Wightman et al. (2021) state that ResNet is one of the most popular networks for image classification. A ResNet, or residual network, is a type of convolutional neural network designed to combat the problem of vanishing gradients in very deep networks. Vanishing gradients occur when the gradients that guide learning get very close to zero when they get back-propagated through many layers. This commonly happens when the network gets deep and it leads to poor training. The ResNet introduced residual learning via skip connections as the solution to this problem.

The core idea of the ResNet is to introduce a residual function rather than for the network to try to learn the full transformation directly. The residual function is the difference between

the desired output and the input. When rewriting that, the output then becomes the sum of the residual function and the input. This addition is made through skip connections. In practice, this means that the original, unmodified, input is carried forward past one or more layers. Meanwhile, on another path, the input goes through several convolutional layers to compute a transformation. Finally, those two, the unchanged input and the transformation, are added together before nonlinear activation. This skip connection ensures that if deeper transformations are not helpful, the network can just pass the input through unchanged. That way it can combat poor performance throughout the depths of the network.

3.3.2 EfficientNet

EfficientNet was first introduced by Tan and Le (2020) in the paper *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks* where it quickly became famous for achieving state-of-the-art accuracy with far fewer parameters and FLOPs than prior models like ResNet. Previous CNN models had a scaling problem, where they had limited computational power and would only scale the network in one dimension, either width, depth or resolution.

EfficientNet combats this with its own type of scaling called the compound scaling method. This method scales all three dimensions at once by using a compound coefficient that is subject to a constraint. The constraint ensures balance between the three different dimensions, scaling them all as evenly as possible. Width, depth and resolution all have different effects on the network. Scaling the width means adding more channels (feature maps) per layer. Each channel can, in rough terms, be seen as a detector for one type of feature in the data, so increasing this would mean finding more patterns in parallel. Mathematically, this is equivalent to increasing the dimension of the representation space, meaning we can encode more information. Increasing depth means stacking more non-linear transformations (that cannot be collapsed into one transformation), which allows for the discovery of more advanced patterns and introduces more abstract reasoning into the network. To scale the resolution means to increase the number of pixels, *i.e* to increase how detailed our input image is. Higher resolution means more input information, since smaller objects become more visible and the image becomes more detailed. All of this improves accuracy in different ways. That is why the EfficientNet compound scaling provides a better accuracy-efficiency trade-off than only scaling one dimension.

EfficientNet has become a popular family of CNNs and are commonly used as benchmarks. It is also still being improved upon. A later version of EfficientNet, EfficientNetV2 has added functionalities such as adaptable regularization, making it even more competitive (Tan and Le, 2021). Adaptable regularization was added to reduce the drop in accuracy caused by the chosen training method of EfficientNetV2. The training method is very efficient and relies on training with a very small image size from the start, and then progressively increasing it. When adaptively adjusting the regularization, for instance by changing the dropout or augmentation, the drop in accuracy can be compensated for. The way it works is that early in training, when images are small, less augmentation (distortion) of the image is needed and dropout can be lighter. This is equivalent to weaker regularization, and it works well here because the small image has low resolution, so the network is focused on learning course features. The risk of overfitting is low because the data is more general than detailed. As images grow larger during training, we want to increase regularization by adding more augmentation

and higher dropout, because this prevents overfitting to the finer details that larger images provide. The result of this is much faster training with no cost in accuracy.

3.4 Transformers

The transformer was introduced by Vaswani et al. (2017) in the paper *Attention Is All You Need*. It quickly became the state-of-the-art for machine translation, surpassing previous RNN- and LSTM-based architectures in both accuracy as well as training efficiency. Nowadays it is the dominant architecture for all NLP tasks, much owing to its robust attention mechanism. During training, sequences of text are split into tokens through a process called tokenization. A token can be an entire word or symbol or it can be a small part of a word, depending on user frequency and length. Each token gets represented as an embedding vector, with a pre-determined size, with the help of a learned embedding matrix. Unlike CNNs, transformers have no built in sense of spatial relationships between input tokens. Therefore, the position of each token is used to create positional embeddings, which represent the token's position within the sequence that it is a part of. The positional embeddings can be learned or sinusoidal. The token embeddings and positional embeddings are added and the result are new, position aware, vector representations of the tokens that are ready to be passed into the encoder.

The encoder consists of six duplicate layers that have two sub-layers respectively. In the first sub-layer, multi-headed self-attention is applied, which means that each sequence applies attention to itself, multiple times in parallel. In this sub-layer, three learned linear projections are applied to each position-aware token vector, creating three new vectors called $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$. These represent query, key and value for each token respectively. Query represents the features of other tokens that are of importance to the current token. Key represent the characteristics of the current token that might be of importance to other tokens. Value is the representation of the actual information that the token holds. The attention process maps the query and the key value pairs of each token to a new vector that is computed through a weighted sum of the values (Vaswani et al., 2017). The weights are determined by a computed compatibility score between the query and key vectors for each token within a sequence. These scores, normalized using softmax, determine how much each token's value vector contribute to the output. The new vector representations have the same shape as before but they are now context-aware. Both sub-layers in the encoder are surrounded with residual connections and these add back the initial input vectors of the layer, before normalization is applied.

The next sub-layer is a position-wise feed-forward network, which takes the output from the previous sub-layer as input. The purpose of this layer is to introduce non-linearity into the model. This is achieved by applying an activation function, such as ReLu, to each token independently. This step is of great significance because it enables the model to learn more complex relationships between the input- and output data. Attention is a linear process; consequently, without the activation function, there would be no benefit in having a deep network since all layers would collapse into one linear transformation. The result are new vectors with the same size as determined by the model dimension and analogously with the previous sub-layer they go through residual connections and normalization. Then they are fed to the next layer, which is identical, and the process repeats through the encoder's six

layers. The final output of the sixth layer is then sent into the decoder.

The decoder has a very similar architecture to the encoder. It has six layers, but each layer has three sub-layers instead of two. The two first sub-layers are identical to the ones inside the encoder with one key difference in the self-attention layer. In the decoder this layer is masked, meaning that the decoder is restricted to only attending to previous outputs and not future ones. This ensures that when the decoder generates a prediction for a token, it can only base it on the tokens it has already predicted. This is because in the decoder, the entire sequence is not known from the start. The third sub-layer performs an additional multi-headed attention, however, not self-attention. Instead, it performs cross-attention which refers to the decoder being able to attend to the outputs of the encoder stack. The final result from the decoder are logits that predict how likely each token in the vocabulary is for a place in each sequence. Finally, detokenization is performed and the predicted text is attained. To achieve the best performance, transformers are typically trained on very large datasets and then finetuned with much smaller datasets in order to become specialized for a certain task. For a visualization of the encoder-decoder structure inside the transformer, see Figure 3.1.

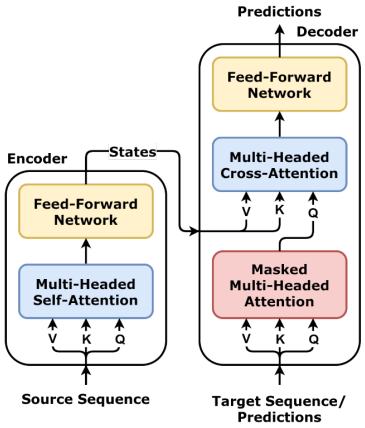


Figure 3.1: An illustration of an encoder-decoder block in the classic NLP transformer (Wikipedia contributors, 2026a).

Even though transformers were primarily developed for NLP tasks, their core attention mechanisms can still be applied to image classification tasks with great success (Bhojanapalli et al., 2021). Since images and text are structurally different, the model architecture differs as well and can vary between different transformer models. The vision transformer (ViT) was introduced in the paper *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale* by Dosovitskiy et al. (2021). Figure 3.2 illustrates the architecture of a general vision transformer. In contrast to the original transformer which it was inspired by, it does not have an encoder-decoder structure. Instead, it has an encoder and a MLP classification head. Nevertheless, the ViT has much in common with the original transformer.

Firstly, there is a tokenization process of the data before it is passed to the encoder. For image datasets, this means that each image gets divided into smaller, non-overlapping patches. Essentially, these patches are equivalent to the tokens used in the NLP transformer. In the self-attention process, this means that each patch attends to all the other patches. Moreover, each patch is flattened into a vector and then embedded by passing through a

learned linear projection. The next step is to add positional embeddings to inject spatial relationships into the model. Finally, a special classification token is added and its purpose is to store the information from the attention layers, acting as a representation of the image as a whole.

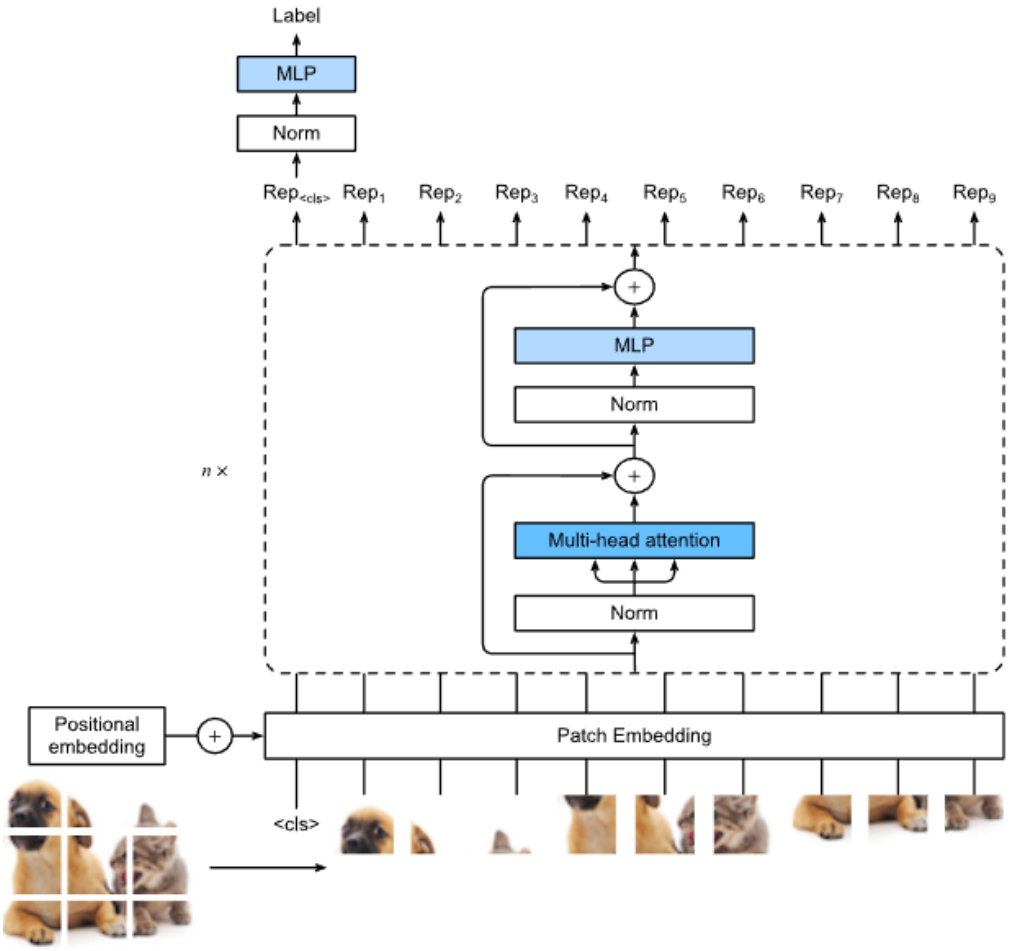


Figure 3.2: A basic visualization of a general vision transformer architecture. The input image (left) is split into patches which are flattened, embedded and combined with positional encodings. These embeddings pass through multiple self-attention layers where each encoded patch interacts with all the others to capture relationships. After the attention layers the generated context-aware feature representations go through normalization and are fed to a classifier head which produces the final predicted label (Wikipedia contributors, 2026b).

The encoder is made up of stacked layers where each layer has two sub-layers, multi-headed self-attention block and a feed-forward MLP block. Layer normalization is applied before each sub-layer and residual connections are applied after. Consistent with the NLP transformer, the vision transformer keeps vectors at a constant dimension throughout all its layers, simplifying vector transformations between them. The MLP layer consists of two linear layers with a nonlinear GELU activation in between, to introduce complexity into the model. After the final output layer, the embedding for the classification token is extracted and fed through a classifier head, which maps the embedding to one of the output classes.

Dosovitskiy et al. (2021) found that a downside of using transformers for image classification is that they lack positional knowledge of patches within an image, something that CNN architectures have naturally. For CNNs, this positional awareness is often referred to as an inductive bias, since it is something that can improve the model's predictions without having to be trained into the model. Transformers do not have this inductive bias. Consequently, all spatial relationships that are to be known by the transformer have to be injected or learned from scratch, which increases its computational demand.

3.4.1 CrossViT

The cross-attention multi-scale vision transformer (CrossViT) was first mentioned in the paper *CrossViT: Cross-Attention Multi-Scale Vision Transformer for Image Classification* by Chen et al. (2021). Its purpose is to build upon the success of ViT by adding multi-scale feature representations to the model. The original ViT (Dosovitskiy et al., 2021) processes images using patches of a fixed size, 16x16. When choosing the granularity of the patch size, there is a trade-off between computational cost and accuracy, since smaller patches can capture more details but increase the number of computations.

Multi-scale feature representations are used to extract information from images in varying detail using multiple patch sizes. For instance, you can zoom in and capture very small features like edges, angles and textures. At the same time, you can zoom out and capture larger characteristics or even the object in its entirety, as well as its position within the image. This will allow the model to see more patterns, because often, different types of characteristics are visible at different scales. An example of this is that a cat's whiskers is a small scale feature whereas its head is a larger scale feature, see Figure 3.3 where the entire CrossViT architecture is visualized.

The CrossViT architecture consists of a stack of multi-scale encoders with two branches per encoder, one with small patches that extracts small scale features and one with larger patches that extract large scale features. As a result of the smaller image patches being heavier to process, the small patch branch contains fewer encoders than the large patch branch, in order to balance the computational load. What is more, each branch gets learnable positional embeddings as well as its own classification token added to their patch embeddings. The branches process the different size patch embeddings separate from each other by independently feeding them through a stack of internal traditional ViT encoder layers (Dosovitskiy et al., 2021), until the branches converge in the cross-attention step.

The transformer encoders perform self-attention on each branch and their respective classification tokens collect information from its own patches through the self-attention process. Then it is time for the cross-attention and at this step, the two branches exchange information. The cross-attention process is performed for each branch, symmetrically. For the small branch, it starts with the computation of the linear projections query ($\boldsymbol{Q}$), key ($\boldsymbol{K}$), and value ($\boldsymbol{V}$). $\boldsymbol{Q}$ is based on the the small branch's class tokens and is a representation of what this token is "looking for" in other tokens. The K projection is based on the large branches patch tokens, and is a representation of how each token describes itself, so that other tokens may decide whether to pay attention to them. The actual information in the patches are represented in $\boldsymbol{V}$ and they are also based on the large branch. When this process is carried out within the large branch, all roles are reversed. By doing this, rather than mixing all patch tokens in both branches for cross-attention, the time complexity of the model is reduced.

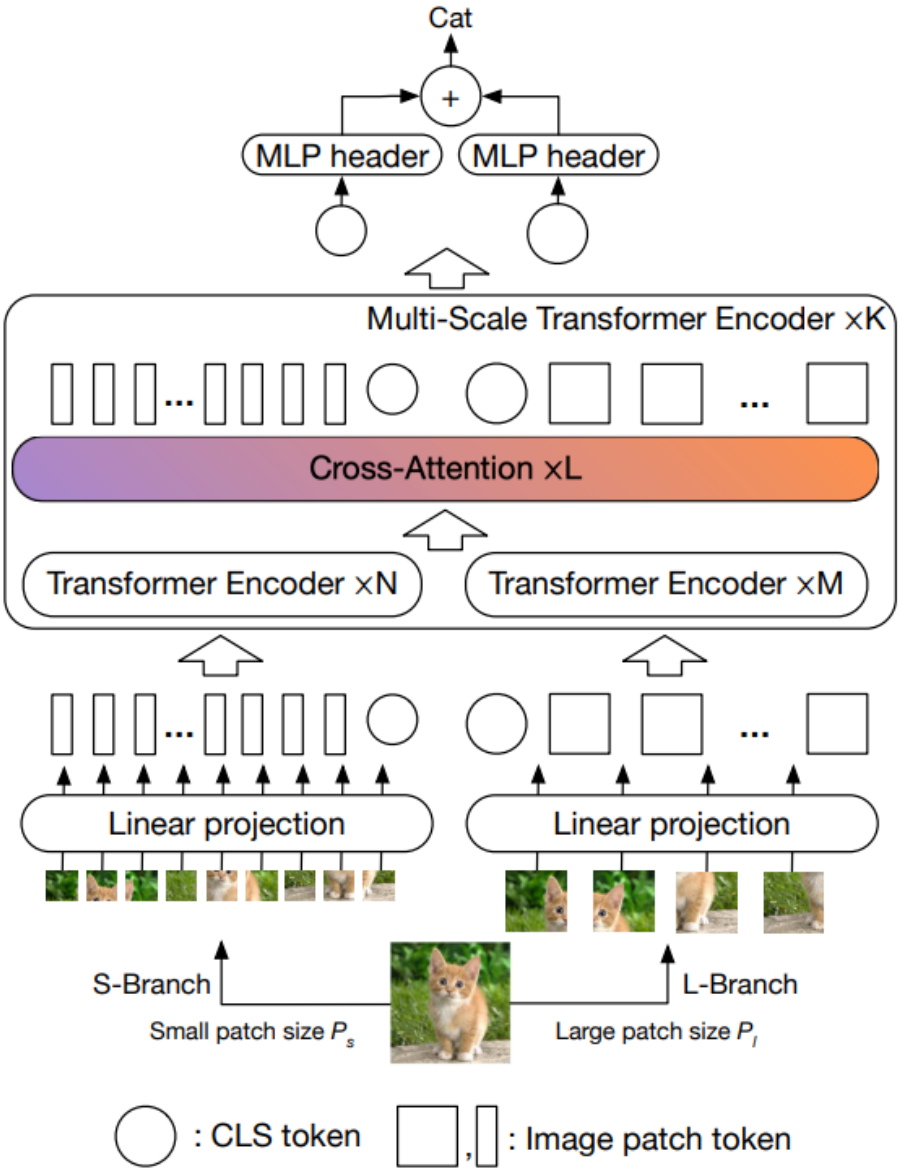


Figure 3.3: The architecture of the CrossViT. Adapted from Chen et al. (2021) and licensed under CC BY-NC-SA 4.0.

The dual branches of the CrossViT introduces the ability to train with small patch sizes without it becoming computationally overbearing. For small images, this approach has much potential.

3.4.2 DeiT

Touvron et al. proposed the data-efficient image transformer (DeiT) in the paper *Training data-efficient image transformers & distillation through attention* (Touvron et al., 2021). It builds upon the original vision transformer architecture by Dosovitskiy et al. (2021) but aims to combat the issue of them not generalizing well unless trained on vast amounts of data. This is achieved by a technique called knowledge distillation (KD), which makes use of a teacher-student architecture. The idea is to have a small, lightweight model be trained to copy the outputs of a much larger, much heavier model. However, it is still possible to train the model without a teacher.

Touvron et al. offer a new approach to knowledge distillation, especially adapted for transformers, where a distillation token is added to the model. This token is added to the initial embeddings (the classification token and the patch embeddings) and it allows the model to learn from the predictions of a chosen teacher model. The standard choice for DeiT is to have the teacher model be a larger DeiT model. This is what the pretrained distilled DeiT uses.

The distillation token is included in the self-attention process and it works as a complement to the class token. Instead of having the image labels as target, the distillation token has its own special target, namely the outputs of the teacher model. The distillation vector and the class vector start out as very different vectors, with cosine similarities in the size range of a few hundredths. As training progresses, they get more and more alike, but never 100 percent similarity. This is because the teacher model is meant to provide additional information to actually be beneficial. If the distillation vector were identical to the student model's class vector, it would be unnecessary.

For the class prediction, the DeiT model has two classifiers, one for the final distillation embedding and one for the final classification embedding, both outputting logits over the classes. The classification token classifier uses the standard cross-entropy loss between the predicted logits and the ground truth class labels, analogously with the original ViT. The distillation token classifier instead uses a cross-entropy loss with the teacher models predicted probabilities as the target. For obtaining the final prediction, Touvron et al. recommend that both classifiers influence the the output. This is done by averaging the two softmax outputs from the two classification heads, thereby predicting the class with the highest combined probability.

The main advantage of using DeiT is that it is more computationally efficient without sacrificing accuracy, by leveraging the results from a larger, already trained model. Another advantage is that the distillation process allows inductive biases to be transferred from the teacher model to the student model. This means that a student transformer model could gain, for instance, some positional awareness if taught by a CNN teacher model. This shows how versatile the DeiT model is when used with different teacher models.

3.4.3 EfficientFormer

In the paper *Rethinking Vision Transformers for MobileNet Size and Speed*, Li et al. (2023) investigate how to adapt the vision transformer architecture to be able to run smoothly on mobile devices with less computational power. This is because they noticed that an improved accuracy with a transformer model often comes with the cost of latency and large model size. Their objective is to come up with a transformer model that can compete with MobileNet performance-wise and still be of a similar size. They also impose constraints on latency, trying to optimize the model's inference time.

Li et al. start with identifying inefficient design choices in previous vision transformers and try to modify them to be more efficient. Consequently, the EfficientFormer architecture has been optimized in several ways. Firstly, the pooling layers are replaced with depth-wise convolutions of the same kernel size, which are placed within the feed forward network. A depth-wise convolution is a convolution that acts separately on each input channel. This is beneficial because it increases accuracy by injecting positional information into the model, without increasing the number of parameters notably.

Secondly, they optimized the search space, which means they re-evaluated the number of stages the network should have, how many blocks each stage should have and how many channels each block should have. They do this to ensure that the network structure is optimal for the new changes that they have made. They found that a deeper but narrower network gave better accuracy, along with a reduction in latency and number of parameters. This means that they added more blocks per stage, but less channels per block. Li et al. also decide against adding a fifth stage and instead keep it at four stages in the network. The reason for this is that, with their other constraints, adding a fifth stage actually resulted in a drop in accuracy and an increase in latency, despite a decrease in FLOPs.

Thirdly, the multi-headed self attention mechanism was improved. In this process, the input is projected onto three matrices, $\boldsymbol{Q}$, $\boldsymbol{V}$, and $\boldsymbol{K}$ just like for the original vision transformer. The difference for the EfficientFormer is that they yet again add a depth-wise 3x3 convolution, right after the formation of the $\boldsymbol{V}$ projection. This is because the multi-headed self attention is a global process where all tokens attend to one another, making the positional information from the added convolution very beneficial. Additionally, they enabled information exchange between attention heads which are usually independent. This was done by adding fully connected layers across head dimensions, called a talking head mechanism. As a result of this, the attention process can capture richer details owing to the collaboration of the multiple attention heads.

Fourthly, Li et al. observed that for high resolution features, the model was not optimal to run on mobile devices due to the quadratic time complexity of the attention mechanism. This is adamant especially in the earlier layers, where resolution is the highest. To combat this, they introduce a stride attention mechanism. From the input feature map, they downsample $\boldsymbol{Q}$, $\boldsymbol{V}$, and $\boldsymbol{K}$ into a fixed lower space resolution (1/32 of the original image resolution) which reduces the number of tokens. When they subsequently perform attention on these downsampled tokens, the time complexity is reduced and at the end, the tokens are upsampled to match the dimensions of the original feature map again.

Lastly, Li et al. introduced dual path attention downsampling into EfficientFormerV2, to improve how spatial resolutions are being reduced for tokens in between stages of the network. In both ViTs and CNNs, spatial resolutions are reduced gradually to allow the model

to capture finer and finer details as features. Traditional approaches to downsamling include pooling and strided convolutions, which are both static and local methods. This is because they have a fixed kernel size and contain no global information. Dual path attention downsampling as used in the EfficientFormerV2 is instead a dynamic and context aware method. The method combines convolution, pooling and attention, therefore containing both local and global information. This approach gave an increase in accuracy with a minimal increase in parameters and latency, a trade-off that the authors deemed worthy.

Li et al. state that they were able to achieve a better performance with their EfficientFormer model than what had previously been achieved by MobileNet. Because of this, its lightweight architecture and its low inference time, it shows great promise for Axis' use case.

3.4.4 FastViT

The FastViT transformer was made by Vasu et al. (2023) with the intention of lowering the computational complexity and memory requirements of the vision transformer. It has a hybrid architecture, combining the strengths of transformers and CNNs to find the best trade-off between accuracy and efficiency. This makes it a suitable choice for deployment on edge or mobile devices. What makes the architecture unique is that it is based on three key design principles.

First, a RepMixer block is used as a token mixer operator. After a token mixing process, such as pooling or self-attention, skip connections are applied and then the result is fed through a feed forward network. However at inference time, these skip connections increase latency. The idea behind the RepMixer is to remove the skip connections, so that they are only present for the training and not the inference of the model.

Second, accuracy is improved with the use of linear train-time over-parameterization. This means that we deliberately use more parameters/branches during training than what is actually necessary. The benefits of this are many. It allows for multiple kernel sizes, which facilitates the capturing of patterns on different scales. Additionally, it enables gradients to flow through different paths, which is much better than having a gradient pass through every layer sequentially. This is because having multiple paths makes the gradient less sensitive and allows it to avoid weak or poorly conditioned paths. By making these effects train-time only, we avoid additional strain on computational resources. During inference, the extra branches are simply collapsed into one operation, which is possible due to the linearity of the extra branches.

Finally, in the early stages throughout the layers of the FastViT, self-attention is replaced by large convolutions with large kernels to reduce computational costs. In the early stages, we have the most tokens because the resolution decreases with depth. Since attention has quadratic time complexity and convolutions are linear, it is much computationally cheaper to only leverage attention in the later stages, when the resolution is lower.

Vasu et al. state that their FastViT transformer is 4.9 times faster than EfficientNet, and 1.9 times faster than ConvNeXt when running on a mobile device, all the while maintaining the same accuracy on the ImageNet dataset.

3.4.5 MaxViT

The Multi-axis Vision Transformer (MaxViT) was introduced by Tu et al. (2022) in the paper with the same name, as a new alternative hybrid transformer with a more simplistic design optimized for scalability. It combines the global strengths of the transformer with the low computational complexity and local bias of convolutions in a new hybrid block which is the core building element that makes up the MaxViT.

MaxViT has its own version of the classic transformer attention mechanism, called Multi-Axis attention. Figure 3.4 illustrates how Multi-Axis attention works. Attention is the foundation of the transformer's success, but as noted by many, it is very computationally heavy due to the operation's quadratic time complexity. In Multi-Axis attention, the full sized attention operation is reduced into a local attention and a global attention by decomposing the spatial axes. This means that every token does not have to attend to all other tokens, like in the classic ViT architecture. Instead, each image is divided into non-overlapping, congruent blocks, creating a grid on the image, and attention is computed within each block. This lets the network learn local patterns and more fine-grained details. In the grid pattern, since all blocks are of the same size, they all contain the same number of tokens and every token has a relative position within each block. This is utilized in the global attention processes, where each token within a block attends to one token from each of the other blocks, namely the token with the equivalent relative position within its own block. Through switching to this attention approach, MaxViT maintains linear computational time complexity.

The architecture of MaxViT is hierarchal and consists of several stages. For each stage, the spatial resolution is lowered and the channel dimension gets increased. A stage is made up of a number of identical MaxViT blocks, which combine multi-axis self-attention with convolutions. Each MaxViT block is divided into two parts. It begins with a convolutional part, which introduces convolutional inductive bias through depthwise separable convolutions and channel-wise attention. This is followed by the attention part, which uses the Multi-Axis attention.

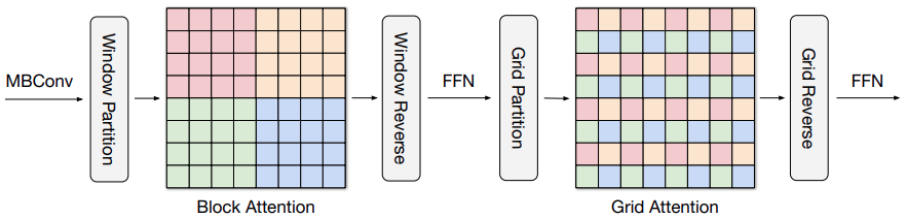


Figure 3.4: How the Multi-Axis attention process in the MaxViT transformer works. Adapted from Tu et al. (2022), licensed under CC BY 4.0 (link: https://creativecommons.org/licenses/by-nc-nd/4.0/).

3.4.6 MobileViT

Mehta and Rastegari (2022) starts their paper *MobileViT: Light-weight, General-purpose, and Mobile-friendly Vision Transformer* by stating that previously, light-weight convolutional neural networks have been the best and most common choice for mobile vision tasks. Their

motivation for this is that CNNs have spacial inductive biases that allow them to learn relationships with fewer parameters. However, they now propose the MobileViT - a transformer that combines the strengths of CNNs and ViTs, while being light-weight enough for mobile vision tasks.

MobileViT has a hybrid CNN and ViT architecture design. In favor of having pure self-attention or pure convolutions, MobileViT uses convolutions for local feature extraction and self-attention to capture global context. First, convolutions are applied and this adds spacial information inductively and detects local features. Then it is time for the attention, but as opposed to the regular ViT, the entire image is not flattened into a single sequence of tokens. Instead, it is carefully divided into patches and then flattened, preserving the local spacial information within each patch. Next, attention is performed between the tokens that have the same relative position within each patch. For instance, all top-left pixels in all patches across the whole image attend to each other. After that, the patches are folded back into an image again, matching the format of a CNN output, *i.e* a height×width×depth feature map. Then, this attention-transformed feature map is fused with the original result of the convolutions in the beginning. This way, original local features are concatenated with global transformer features. Then, an additional convolution is applied to mix these features. This process is the fundamentals of the MobileViT block, the main pillar of the MobileViT architecture. A MobileViT network commonly has 3 stacked MobileViT blocks, which appears at progressively lower resolutions, combined with standard convolutional blocks before, between and after them (Mehta and Rastegari, 2022).

3.4.7 Pyramid vision transformer

The pyramid vision transformer (PVT) was first presented in the paper *Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions* by Wang et al. (2021). The PVT is a convolution-free (pure transformer) architecture that utilizes the pyramid approach commonly used in CNN backbones. Rather than processing an image at a single, fixed scale, like the original ViT, PVT generates feature maps at progressively lower resolution. In the early stages, resolution is at the highest and we have the most tokens. In the later stages, resolution is lower and the amount of tokens is less. By now, we know that attention is a computationally expensive operation, scaling quadratically with the number of tokens. The PVT handles this by introducing spacial reduction attention when learning high-resolution features.

Spacial reduction attention works by down-sampling the key and value tokens to a lower resolution before attention is computed. This reduces both computation and memory requirements. The query tokens remain intact, maintaining the global receptive field that transformers usually have. Attention is then computed in the same way as in the classic ViT. However, instead of producing a single output feature map like ViT, PVT produces a set of feature maps at different scales. The final prediction can then be made by pooling and concatenating the different feature maps.

3.4.8 Swin

Liu et al. (2021) introduced the Swin transformer in the paper *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows* to further adapt the transformer to the image domain.

They believe the greatest challenge for the vision transformer is the large scale of the the visual input tokens compared with textual tokens. Images are more complex, having many pixels and often high resolution. The Swin transformer aims to combat this by utilizing hierarchal features, which are created through down-sampling, *i.e*, merging of patches/tokens, so that each patch gets larger the deeper into the network they go. This reduces the number of tokens each stage and therefore also the computational resources needed by the transformer.

In the attention process, the Swin transformer replaces the regular multi-head attention method with the shifted window approach, which it also is named after (**S**hifted **win**dows). In the attention process of the Swin model, each image is first split into patches in the same way as in a regular vision transformer. Each patch is considered a token. Then, the image is divided into non-overlapping shifted windows. Self-attention is then performed by the patches within each window. The shifted window approach is competitive because it has proven to be lower in latency than previous sliding window methods, without increasing computational load. Then the windows shift, creating new windows, see Figure 3.5. Self attention is performed within these new windows and cross attention with neighboring windows in the previous layer is performed as well. This continues through several stages, where the patch size gets larger for each stage. See Figure 3.6 for a visualization of the Swin architecture. Other than the down-sampling and the shifted window attention, the layers in the Swin transformer are identical to those of the original vision transformer.

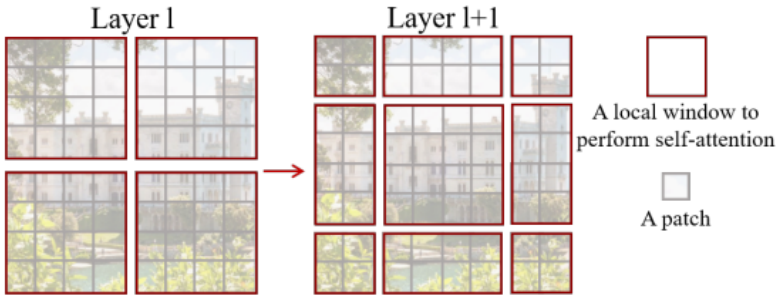


Figure 3.5: How the shifted windows in the Swin transformer work. Adapted from Liu et al. (2021), licensed under CC BY 4.0.

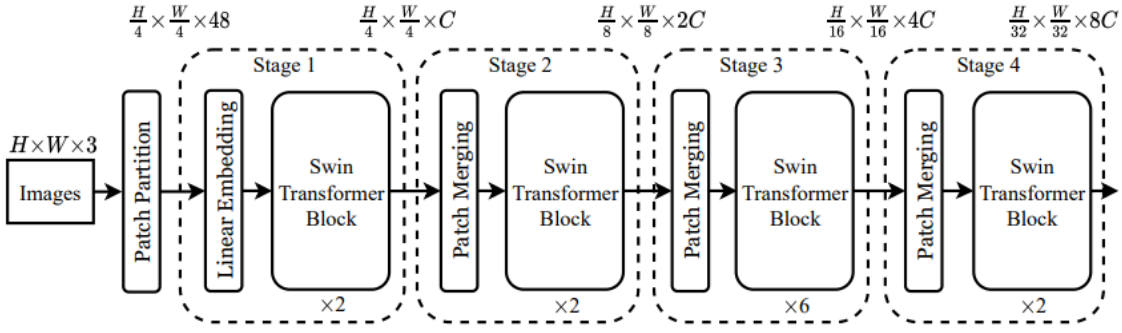


Figure 3.6: Illustration of the Swin Transformer architecture. Adapted from Liu et al. (2021), licensed under CC BY 4.0.

Chapter 4
Methodology

We organized the experiment in the following way:

- Initial model selection
- Finetuning pretrained models
- Model selection for optimization
- Optimizing the architectures
- Training from scratch or with distillation
- Finetuning the new models
- Testing on two datasets
- Further finetuning on dataset of animals

4.1 Initial model selection

When selecting the models to study, it was important to consider model size, number of parameters, model efficiency, inference time and the resolution of the images that the model had been pretrained on. The included models were chosen from a wide set of commonly used transformer models and selected based on their architectures and design. This research resulted in the initial list of transformer architectures to study and they are described in chapter 3. The choice is strongly dependent on number of citations, open-source availability and architecture design.

When selecting a version of the pretrained open-source models from Huggingface, we tried to select versions that had been pretrained on image sizes that were as similar as possible to the ones in Axis' own databases. All image sizes used in this study are smaller than

384, because our use-case specifically concerns small images. We cannot disclose the sizes in greater detail; in our result tables we simply provide whether it matches the input size of Axis' current classifier or whether images have been enlarged to match the input size of a pretrained model.

We want a lightweight model, with low inference time that can be embedded into an Axis camera. The classifier is to be used for small images, and therefore, choosing models that have been trained on smaller images is desirable. Additionally, the transformer models have to be able to be compared as fairly as possible to the current classifier and that means finding models where the number of parameters and the number of FLOPs are in the same magnitude, preferably as close as possible. However, it is a known fact that transformers generally are more computationally demanding than CNNs, therefore we expect the transformers to be heavier. Ideally we try to choose small versions of the models whenever they exist, but we have also included a few larger models in the experiment. This is because we have the opportunity to experiment with the layer architectures and model parameters to reduce model size later, if a model shows initial promise. We also keep in mind that the future might allow for the integration of larger models.

4.1.1 Baseline comparison of pretrained models

The first part of the experiment is divided into two parts. In one part, we compare our selected transformer models with Axis' baseline CNN model. Here we use pretrained transformer models that we finetune on Axis' large dataset. In the other part, we compare our selected transformer models with two other benchmark CNNs, ResNet and EfficientNet. We finetune and test all models on the public FLIR dataset.

For the finetuning during both parts, we start with the weights of the pretrained model and do additional training on our own dataset without freezing any layers. We keep the hyperparameters as similar to what Axis used for their baseline for a fair comparison. However we do adjust the learning rate to be suitable for finetuning, making it smaller.

4.1.2 Model selection for optimization

When choosing the models to optimize further, we based the decision on how the pretrained models performed and which architectures we felt were most promising. DeiT and MobileViT were chosen for their specific optimization for data-efficiency and lightweightness. CrossViT was chosen for its dual-branches approach and performance on the initial experiment. Finally, Swin was chosen because it performed the best on the initial experiment. Other models were either too similar, performed worse or were not as interesting for our use case based on our subjective opinion.

4.1.3 Optimization

In order to further adapt the models to our use case, we will try to optimize the architectures for a few of our favorite models. Then, these selected transformers will be trained from scratch or with distillation. There are many benefits of customizing the architecture of the models we like the most. Primarily, the pretrained models available to us have all been trained

in the color domain whereas our thermal images are in the single-channel domain. Secondly, the resolution of our images is very low and therefore it would be beneficial to be able to reduce both the patch size but also the input image size of our chosen transformer networks. Some networks allow this to some extent even for the pretrained version, but most do not. Lastly, optimizing the architectures will allow us reduce the number of model parameters and the number of FLOPs.

For experiments with optimization both for model architectures and training parameters, we use Optuna[1] (Akiba et al., 2019). Optuna is an open-source tool for optimization. It tries different parameters according to your inputs, and then prunes the search space using all previous results that it has achieved. So, if at a certain point a training with a chosen set of parameters is performing worse than the mean of all previous trainings, Optuna will cancel the training. This saves both time and resources and it gives great visualization of which parameters are good choices when trying to maximize the accuracy.

4.1.4 Training from scratch

When training from scratch we have chosen to use the ImageNet-1k[2] dataset because it is very large (1.2 million images for training) and popular to use in image classification tasks. We alter this dataset to our preferred size and number of channels, and we also crop the images so that only one object may be present in each image. Training from scratch means that we start training with randomly initialized weights. After training on ImageNet-1k, we follow up with finetuning on Axis' large dataset. For this training, we chose our best model architectures from the Optuna trials, based on accuracy and number of FLOPs.

4.1.5 Training with distillation

Training with distillation means incorporating a larger teacher model and letting a smaller student model learn from its predictions. When training with distillation, a special distillation loss is used. As opposed to the usual loss function, the goal is not simply to minimize the difference between the output and the true label. Instead, distillation loss aims to minimize the difference between the teacher's output and the student's output, as well as minimizing the loss between the students output and the true label (Xu et al., 2023).

Xu et al. discuss different types of distillation in the paper *Teacher-student collaborative knowledge distillation for image classification* and reference many different variations with varying degree of success. Distillation can be more or less advanced, and the approach above is on the simpler side. It is possible for the student to extract even more information from the teacher, by not only leveraging its outputs, but also intercepting earlier layers. An example can be letting the student learn from the teacher's hidden layers. In this work, we mainly focus on the basic distillation approach.

[1] `https://optuna.org/`
[2] `https://huggingface.co/datasets/ILSVRC/imagenet-1k`

4.1.6 Testing on an unseen dataset

For the final evaluation of our selected transformer models, we will test them on another one of Axis' confidential datasets, introduced in section 2.1.2. It mainly contains images of large animals such as rhinos, deer and elephants. First, we will analyze the model performance when the test set is completely new and uncorrelated to the training data. The transformers will be compared with Axis original classifier on the same test. Then, we will introduce the training set of large animals and further finetune our models on it. The finetuning will proceed just as before, starting out with the models we have, freezing no layer and simply further training on the new data. Afterwards, we will compare the results on the test set of big animals in the same way as before. This will show the capacity of the different models to be able to distinguish people from large animals, both when specifically finetuned on them and when not.

Chapter 5
Evaluation

5.1 Experimental setup

The experiment is divided into three parts. In the first part, selected transformer models are evaluated against Axis' baseline model. This is done by selecting relevant pretrained transformer models and finetuning them on Axis' own dataset that contains people, crawling people, vehicles, bikes and animals. Table 5.1 shows the detailed names of the chosen pretrained models. Then, these models are finetuned and compared with one another. After that, the most suitable transformer models are selected and further optimized to the task of classifying small thermal images. This includes training from scratch or with distillation, and tuning the model architectures using Optuna. Finally, the models are tested on an unseen dataset of large animals, namely Axis' small dataset, as referenced in section 2.1.2.

When a suitable architecture is found for each selected model, the models are either trained from scratch using ImageNet or trained with distillation. When training with distillation we will use larger successful models (both pretrained and trained from scratch by us) to train a smaller student model. When training from scratch with ImageNet, we convert the images to grayscale and our preferred smaller image size. After that, we finetune our models again on Axis' training dataset. To test these models, we will do a final evaluation of their performance using an unseen dataset of animals. Animals are one of the most common causes of false alarms for the person class, and through this test we will see if any our selected transformer models are actually better at preventing false alarms than Axis' original model.

5.2 Metrics

The metrics considered for the evaluation of the models in this report are the overall test accuracy, recall and precision of the person class, number of model parameters and number of Floating-point Operations (FLOPs). The FLOP gives an estimate of a model's computational

Table 5.1: Model overview with full names and pretraining datasets.

Model	Full name	Pretrained on
ResNet-10-T (tiny)	`resnet10t.c3_in1k`	ImageNet-1k
EfficientNet	`tf_EfficientNet_b0.in1k`	ImageNet-1k
CrossViT	`crossvit_9_240.in1k`	ImageNet-1k
DeiT	`facebook/deit-tiny-patch16-224`	ImageNet-1k
Distilled DeiT	`facebook/deit-tiny-distilled-patch16-224`	ImageNet-1k
EfficientFormerV2	`efficientformerv2_s0.snap_dist_in1k`	ImageNet-1k
FastViT	`timm/fastvit_t8.apple_in1k`	ImageNet-1k
MaxViT	`timm/maxvit_tiny_tf_224.in1k`	ImageNet-1k
MobileViT	`MobileViT_050.cvnets_in1k`	ImageNet-1k
PVT	`Xrenya/pvt-tiny-224`	ImageNet-1k
Swin	`Swin_tiny_patch4_window7_224.ms_in22k_ft_in1k`	ImageNet-1k

cost by computing how many floating point operations it needs to complete for one forward pass, and this in turn can give an idea of its inference time, because generally, lower FLOP entails shorter inference time. For the tests on Axis' small dataset of animals, we will also consider the recall of the animal class. Many results will be presented with confusion matrices, giving a great overview of precision and recall of all classes.

5.3 Results

5.3.1 Comparison of pretrained transformers with baseline

The results of the pretrained transformers are presented in two tables. Table 5.2 shows how the pretrained models perform when finetuned and tested of the public FLIR dataset and table 5.3 shows the corresponding training parameters used. Table 5.4 shows the performance of the pretrained models when finetuned and tested on Axis' large dataset and 5.5 shows the corresponding training parameters used.

Table 5.2: Results on the public FLIR dataset, test set. Baseline comparison.

Model	# params. (M)	Best acc.	Person class			Approx. FLOPs (M)
			R	P	F1	
EfficientNet	5.87	99.351	**99.6**	98.87	99.23	80.92
ResNet-10-T (tiny)	4.93	99.299	99.4	98.85	99.12	113.81
CrossViT	8.17	**99.494**	99.4	**99.30**	**99.35**	3081.47
DeiT	5.53	99.370	99.4	99.04	99.22	2151.21
Distilled DeiT	5.53	99.319	**99.6**	98.70	99.15	2161.84
EfficientFormer	3.25	99.306	**99.6**	98.67	99.13	805.11
FastViT	3.26	99.306	99.3	98.95	99.12	56.31
MaxViT	14.94	99.020	99.0	98.65	98.82	8487.84
MobileViT	1.11	99.026	99.3	98.35	98.82	40.24
PVT	12.72	99.305	99.5	98.82	99.16	3871.96
Swin	27.52	99.461	99.4	99.23	99.31	8994.43

Table 5.3: Model parameters and training accuracy of the models in the baseline experiment using the public FLIR dataset. All pretrained and then finetuned with batch size 256 for 200 epochs, with zero dropout and zero frozen layers.

Model	Learning rate	Training accuracy	Input channels	Image size
EfficientNet	0.0001	99.999	3	small
ResNet-10-T (tiny)	0.00001	99.998	3	small
CrossViT	0.0001	99.602	3	small
DeiT	0.00001	100	3	large
Distilled DeiT	0.00001	100	3	large
EfficientFormer	0.0001	99.998	3	large
FastViT	0.0001	99.990	3	small
MaxViT	0.0001	99.974	3	very large
MobileViT	0.0001	99.999	3	small
PVT	0.00001	100	3	large
Swin	0.00001	99.978	3	large

5.3.2 Selected models

Based on the previous results, the models we select for further optimization are: CrossViT, DeiT, MaxViT, MobileViT and Swin. For the CNN baseline, we choose EfficientNet. These models will be optimized using Optuna and then trained using two approaches; from scratch using ImageNet-1k or with distillation. Finally the models will be finetuned on Axis' large

Table 5.4: Results on Axis' own, non-public large test dataset. Baseline comparison.

Model	# params. (M)	Best acc.	Person class			Approx. FLOPs (M)
			R	P	F1	
Axis	confidential	95.666	97.1	97.00	97.01	143.82
EfficientNet	5.87	90.934	93.4	95.48	94.43	80.92
ResNet-10-T (tiny)	4.93	90.979	92.6	95.58	94.07	113.81
CrossViT	8.17	97.610	98.9	97.50	98.20	3081.47
DeiT	5.53	96.705	98.2	97.10	97.65	2151.21
Distilled DeiT	5.53	97.145	96.9	98.52	97.70	2161.84
EfficientFormer	3.25	97.034	97.0	98.50	97.74	805.11
FastViT	3.26	97.019	97.2	98.74	97.96	282.76
MaxViT	14.94	97.998	98.8	98.23	98.51	8487.84
MobileViT	1.11	96.112	97.6	95.98	96.78	943.74
PVT	12.72	96.381	97.9	97.19	97.54	3871.96
Swin	27.52	**98.455**	**99.2**	**99.10**	**99.15**	8994.43

dataset from section 2.1.1 and tested on the corresponding test dataset and the dataset containing only large animals from section 2.1.2. At the end, we will also finetune on the train split of the large animals dataset, and test it on the large animals test set again.

5.3.3 Optimization with Optuna

Next, we optimize our selected transformer models using Optuna. This is done in order to find the best architecture of our selected transformer models, with regards to FLOPs and accuracy. We only test architectures that are smaller or equal to the pretrained transformers in size. The goal is to minimize our most promising transformer models so that they are small enough to run on an embedded system.

Optuna performed several trainings with our different suggestions for parameter values and in the end we chose the parameters that gave the best trade-off between accuracy and FLOPs. Something noteworthy is that for the Optuna trials, the accuracy will be lower than what we can usually achieve, and this is mainly because of two reasons. Firstly, we do not train for as many epochs as we usually do, and secondly, we train with randomly initialized weights (*i.e* from scratch) but with much less data then what the models were originally pretrained on. This is because we use only the train split of Axis' large dataset, which we normally only finetune with. We deliberately chose to do this to conserve computational resources. The point of these Optuna experiments is not to get a perfect model at the end, just to get an idea of what some good architecture choices would be for each model. Then, in the next step, the model versions with the best architectures will be trained thoroughly.

Table 5.5: Model parameters and training accuracy of the models in the baseline experiment using Axis' large test dataset.

Model	Learning rate	Training accuracy	Input channels	Image size
Axis' current classifier	0.1	99.725	1	small
EfficientNet	0.0001	99.966	3	small
ResNet-10-T (tiny)	0.00001	99.950	3	small
CrossViT	0.0001	99.958	3	small
DeiT	0.00001	99.960	3	large
Distilled DeiT	0.00001	99.962	3	large
EfficientFormer	0.0001	99.984	3	large
FastViT	0.0001	99.979	3	medium
MaxViT	0.0001	99.975	3	very large
MobileViT	0.0001	99.986	3	large
PVT	0.00001	99.957	3	large
Swin	0.00001	99.961	3	large

CrossViT

We started with some basic experiments with the CrossViT transformer. We first tried with and without converting the images to grayscale, the result of this experiment can be seen in Figure 5.1. To reduce the number of experiments that we had to do, we decided to always convert the images we train and finetune on to grayscale, based on this result. Then we experimented with the architecture parameters that are specific to the CrossViT transformer, namely the small patch size, the large patch size, the small embedding dimension and the large embedding dimension. We found the best hyperparameters for CrossViT out of the ones in our search scope, but for confidentiality reasons they cannot be included in this report. The best custom CrossViT model has 132 M FLOPs. In addition, in order to save time and because other transformers seemed more promising, we did not do any further experiments with CrossViT, instead we found that using it as a teacher when training with distillation gave better results.

EfficientNet

Since EfficientNet already had the lowest FLOP of only 80.92 million, we did not experiment to further reduce it.

DeiT

Then we continued on with the non-distilled DeiT transformer. For DeiT, the parameters we experimented with were patch size, hidden size, number of hidden layers, number of attention heads, and intermediate size. We found the best parameters for DeiT, but for con-

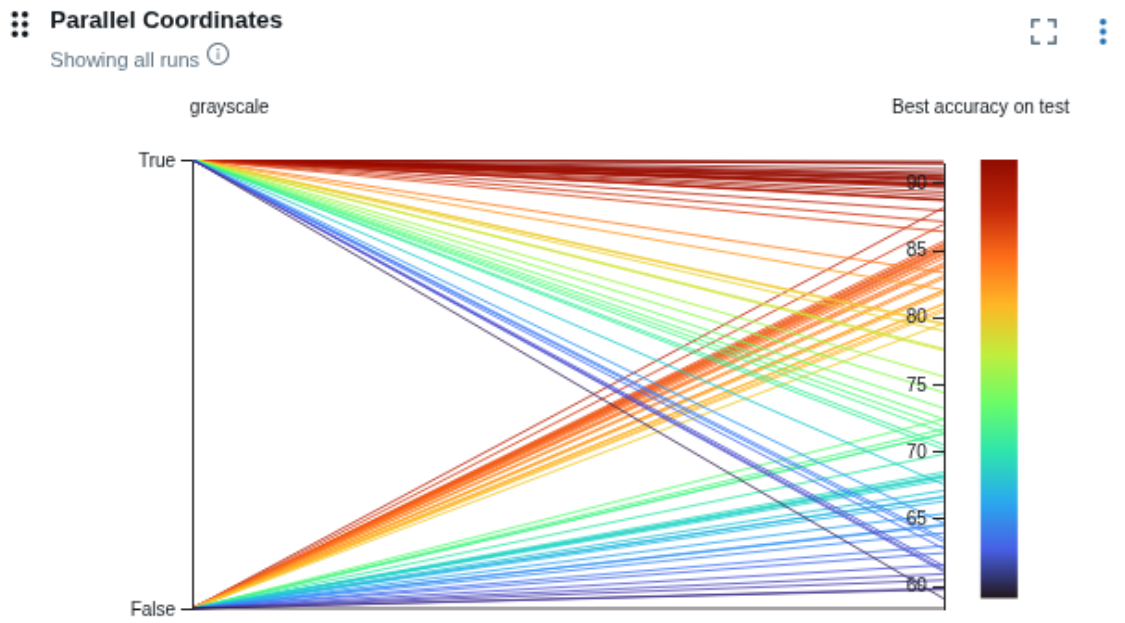


Figure 5.1: The graph shows all trials with Optuna for the CrossViT transformer. By looking at which option, true or false, that produced the most red lines, we can clearly see that using grayscale images improves accuracy.

fidentiality reasons they may not be published in this report. Our new, custom DeiT model has around 777 M FLOPs.

MaxViT

We found the best result of the Optuna trial with MaxViT, but for privacy reasons they may not be published in this report. The parameters tested were channels and grid window size. The new, custom MaxViT model has around 136 M FLOPs.

MobileViT

For MobileViT, we experimented with output stride, attention unit dimension, number of attention blocks, filter sizes and number of MobileViT blocks. We found the best architecture choice for MobileViT but for privacy reasons these results cannot be published. Our custom MobileViT model has 136 M FLOPs.

Swin

For the Swin transformer, we experimented with embedding dimension, patch size, window size, depths and number of heads. The result of this trial can be seen in Figure 5.2 and in Table5.6. We also did an additional Optuna trial with Swin and the result of that can be seen in Figure 5.3 and Table 5.7. We chose to include these results since this model did not end up being our best one, and to give an idea of what the results of these experiments looked like.

We selected the architecture in Table 5.7 for further experiments because it had a good trade-off between FLOPs and accuracy. The architecture has only about 200 M FLOPs which

is very little when compared with the original pretrained tiny patch Swin model, which has over 8000 M FLOPs.

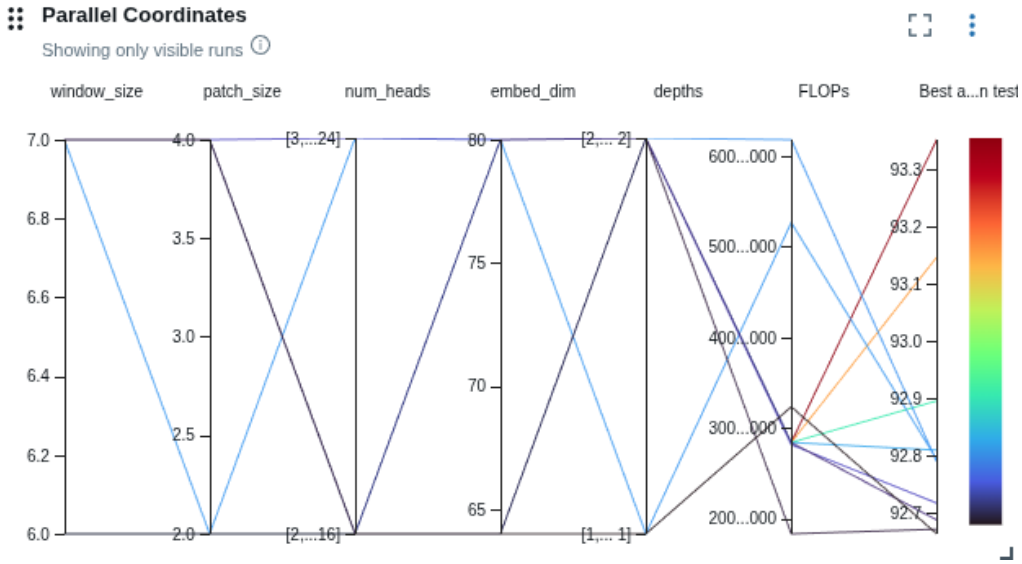


Figure 5.2: The result of the first Optuna trial with Swin. We can see how the architecture choices relate to the number of FLOPs. The graph shows the 10 best trials.

Table 5.6: Best hyperparameters for Swin from the first Optuna trial.

Parameter	Value
embed_dim	80
depths	[2, 2, 4, 2]
num_heads	[2, 4, 8, 16]
window_size	7
patch_size	4
best_accuracy_on_test	93.35
FLOPs	284.0 M

Table 5.7: Best hyperparameters for Swin from the second Optuna trial.

Parameter	Value
embed_dim	72
depths	[2, 2, 3, 2]
num_heads	[1, 2, 4, 8]
window_size	3
patch_size	4
best_accuracy_on_test	93.054
FLOPs	202.6 M

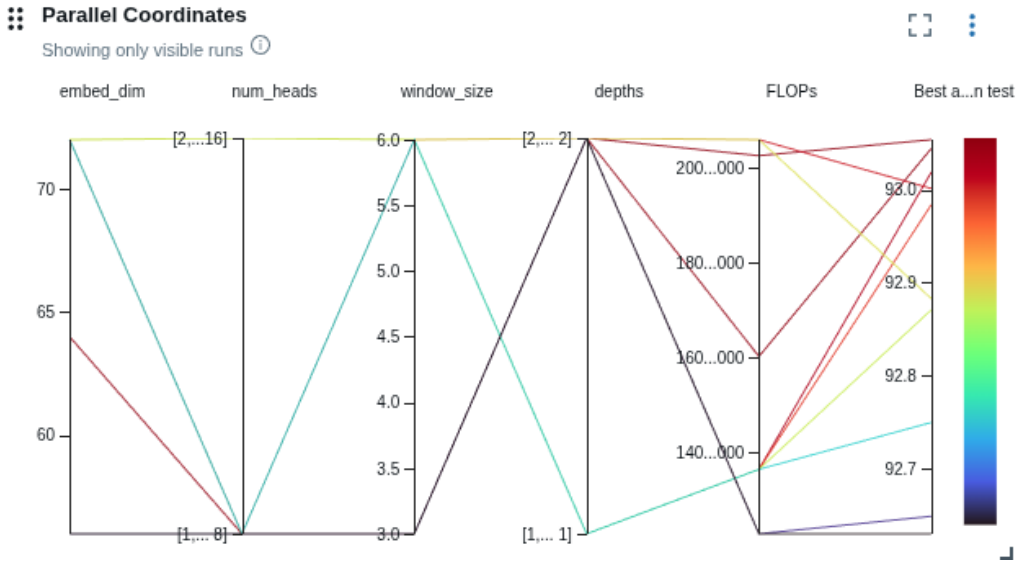


Figure 5.3: The result of the second Optuna trial with Swin. We can see how the parameter choices affect the FLOP. The graph shows the 10 best trials.

5.3.4 Training from scratch with ImageNet-1k

When training from scratch, the following training hyperparameters that can be seen in Table 5.8 were used, for all trainings. The finetuning was made in the exact same manner as for the pretrained models, with a batch size of 256 for 200 epochs, with a much smaller learning rate than for training and freezing no layers. The results are presented in the form of confusion matrices generated from Axis' large test set and Axis' small test set of animals. The results are also summarized in three tables, table 5.9 shows the recall of the person class on Axis' large training set, table 5.10 shows the recall of the animal (false alarm) class on Axis' small dataset and table 5.11 shows the overall accuracy of the models.

Table 5.8: Training hyperparameters used for training from scratch on ImageNet-1k.

Parameter	Value
Optimizer	AdamW
Epochs	300
Learning rate schedule	Cosine decay
Warm-up	5 epochs (linear)
Batch size	1024
Initial learning rate	0.001
Weight decay	0.05

Table 5.9: Summary of results on Axis' large test dataset. Training from scratch and finetuning on Axis' large training dataset.

Model	Recall person class
Axis (original)	97.1
DeiT	**98.1**
EfficientNet	96.8
MaxViT	97.0
MobileViT	**98.1**
Swin	97.7

Table 5.10: Summary of results on Axis' small test dataset of animals and vehicles. Training from scratch and finetuning on Axis' large training dataset.

Model	Recall animal class
Axis (original)	57.0
DeiT	**89.5**
EfficientNet	80.0
MaxViT	54.7
MobileViT	68.0
Swin	79.7

Table 5.11: Overall accuracy of the models when trained from scratch on ImageNet-1k and finetuned on Axis' large training dataset.

Model	Overall accuracy
Axis (original)	95.7
DeiT	93.0
EfficientNet	86.7
MaxViT	**96.0**
MobileViT	95.3
Swin	90.5

Axis' original classifier

The confusion matrix resulting from training Axis' original classifier from scratch only on Axis' large dataset (not on ImageNet-1k), and testing on Axis' original test set can be found in figure 1a. The result of that same model but on the unseen test set of large animals is presented in figure 1b.

DeiT

The confusion matrix resulting from training DeiT from scratch, finetuning and testing on Axis' large dataset can be found in figure 1c. The result of that same model but on the unseen test set of large animals is presented in figure 1d.

EfficientNet

The confusion matrix resulting from training EfficientNet from scratch, finetuning and testing on Axis' large dataset can be found in figure 2a. The result of that same model but on the unseen test set of large animals is presented in figure 2b.

MaxViT

The confusion matrix resulting from training MaxViT from scratch, finetuning and testing on Axis' large dataset can be found in figure 2c. The result of that same model but on the unseen test set of large animals is presented in figure 2d.

MobileViT

The confusion matrix resulting from training MobileViT from scratch, finetuning and testing on Axis' large dataset can be found in figure 3a. The confusion matrix of that same model but tested on the unseen dataset of large animals can be found in figure 3b.

Swin

The confusion matrix resulting from training Swin from scratch, finetuning and testing on Axis' large dataset can be found in figure 3c. The confusion matrix of that same model but tested on the unseen dataset of large animals can be found in figure 3d.

5.3.5 Training with distillation

Below are the results of training each model with distillation using either a MaxViT or a CrossViT teacher model. The CrossViT teacher has 3079 M FLOPs and the MaxViT teacher has 10778 M FLOPs, and they are much larger than our student models. We chose these as teacher models because they were two of our largest models and their performance on the first experiment with the non-optimized models were very good. Additionally, when studying their architectures, we liked their features the best. For CrossViT, the dual branches approach that finds both small and large scale features seemed promising. For MaxViT, the hybrid approach combining convolutions for local pattern-finding and attention for global pattern finding made us want to leverage it as a teacher model. Our reasoning was that the more complex the teacher model was, the more the student model would learn from it. We would have liked to experiment with additional teacher models, but there was not enough time. The results for each model are presented in the form of confusion matrices for the two different test datasets, Axis' large test set and Axis' small dataset of animals and vehicles. The results from this section are summarized in tables 5.12, 5.13 and 5.14.

Table 5.12: Summary of model results when trained with distillation on Axis' large test dataset.

Model	Teacher	Recall person class
DeiT	CrossViT	96.1
EfficientNet	CrossViT	93.6
MaxViT	CrossViT	**98.5**
MobileViT	CrossViT	97.8
MobileViT	MaxViT	97.2
Swin	MaxViT	51.1

Table 5.13: Summary of model results when trained with distillation and tested on Axis' small test dataset of animals and vehicles.

Model	Teacher	Recall animal class
DeiT	CrossViT	76.3
EfficientNet	CrossViT	**79.8**
MaxViT	CrossViT	51.1
MobileViT	CrossViT	56.6
MobileViT	MaxViT	79.6
Swin	MaxViT	70.2

Table 5.14: Overall accuracy of the models when trained with distillation on Axis' large training dataset.

Model	Teacher	Overall accuracy
DeiT	CrossViT	84.06
EfficientNet	CrossViT	87.75
MaxViT	CrossViT	**97.28**
MobileViT	CrossViT	95.59
MobileViT	MaxViT	94.75
Swin	MaxViT	40.12

DeiT

Table 4a and 4b show the results of training DeiT with distillation using a CrossViT teacher model.

EfficientNet

The results of training EfficientNet with distillation using a CrossViT teacher model can be found in Figures 4c and 4d.

MaxViT

The results of training MaxViT with distillation using a CrossViT teacher model can be found in figures 5a and 5b.

MobileViT

The results of training MobileViT with distillation using a CrossViT teacher model can be found in figures 5c and 5d. The result of training MobileViT with distillation using MaxViT as the teacher model is presented in figures 6a and 6b.

Swin

Figures 6c and 6d show the result of training Swin with distillation using a MaxViT teacher model.

5.3.6 Finetuning on animals

For the final evaluation of our models, we will take our models that were trained from scratch on ImageNet-1k and then finetuned on Axis' large dataset and further finetune them on the train split of Axis' small dataset of animals and vehicles. After that we test again on Axis' large test set and on Axis' small dataset of animals. We will also include our favorite distilled model, MaxViT distilled with CrossViT, in this experiment. Table 5.15, 5.16 and 5.17 summarize all of the results from this section.

Table 5.15: Summary of model results on Axis' large dataset after finetuning on Axis' small dataset of animals.

Model	Recall person class	Precision person class
Axis' original classifier	97.9	96.9
DeiT	98.2	96.5
EfficientNet	95.0	94.7
MaxViT	98.0	97.18
MaxViT dist. w. CrossViT	**98.7**	94.9
MobileViT	96.6	95.8
Swin	97.2	**97.3**

Axis' original classifier

Figures 7a and 7b show the result of finetuning Axis' original classifier on the train split of the small dataset of animals and vehicles.

Table 5.16: Summary of model results on Axis' small dataset of animals after finetuning on the train split of that dataset.

Model	Recall animal class
Axis' original classifier	96.5
DeiT	**98.2**
EfficientNet	95.2
MaxViT	97.1
MaxViT dist. w. CrossViT	95.7
MobileViT	86.7
Swin	95.8

Table 5.17: Overall accuracy of the models after further finetuning on Axis' small dataset of animals and vehicles.

Model	Overall accuracy
Axis' original classifier	95.15
DeiT	96.17
EfficientNet	91.12
MaxViT	96.31
MaxViT dist. w. CrossViT	**97.28**
MobileViT	94.38
Swin	94.88

DeiT

Figures 7c and 7d show the result of further finetuning DeiT on the training split of the small dataset of animals and vehicles, after having trained from scratch on ImageNet-1k and finetuned on Axis' large dataset.

EfficientNet

Figures 8a and 8b show the result of further finetuning EfficientNet on the training split of the small dataset of animals and vehicles, after having trained from scratch on ImageNet-1k and finetuned on Axis' large dataset.

MaxViT

Figures 8c and 8d show the result of further finetuning MaxViT on the training split of the small dataset of animals and vehicles, after having trained from scratch on ImageNet-1k and finetuned on Axis' large dataset. The result of training MaxViT with distillation using a CrossViT teacher and then finetuning further on Axis' small dataset of animals can be seen in figures 9a and 9b for the respective test sets.

MobileViT

Figures 9c and 9d show the result of further finetuning MobileViT on the training split of the small dataset of animals and vehicles, after having trained from scratch on ImageNet-1k and finetuned on Axis' large dataset.

Swin

Figures 10a and 10b show the result of further finetuning Swin on the training split of the small dataset of animals and vehicles, after having trained from scratch on ImageNet-1k and finetuned on Axis' large dataset.

5.4 Discussion

In this section we will discuss the results, future work and answer the research questions:

- Can a transformer model outperform Axis' original classifier while scaled down to similar size?
- What training method is the most successful for classification of small thermal images, finetuning or distillation?
- Is a transformer-based architecture a better choice than a CNN for classification of small thermal images under size constraints?

5.4.1 Discussion of research questions

The first step of our experimental evaluation was to finetune pretrained transformer models on Axis's large training dataset, and separately, on the public FLIR dataset. These models were then evaluated on Axis's large test set and the FLIR test split, respectively. On Axis's test set, Swin outperformed all other models across all evaluation metrics. However, Swin is also by far the largest model, with over 27 million parameters and nearly 9,000 M FLOPs, making its superior performance less surprising. Nevertheless, this result provided valuable insight into which architectures showed the most promise for our use case and informed our selection of models for further experimentation, namely DeiT, CrossViT, MaxViT, MobileViT, and Swin. EfficientNet was also chosen as the CNN baseline.

The FLIR test showed different results, with CrossViT being the best in all categories but one, the recall of the person class. This metric was won by two transformers, distilled DeiT and EfficientFormer, along with one of our baseline CNN models, EfficientNet. Across both datasets, performance values were extremely high and often close to 100 percent, especially for the FLIR results. When examining training accuracy, a similar pattern emerges: several models achieved nearly perfect performance, and three transformers trained on FLIR reached exactly 100 percent training accuracy. This strongly suggests that the models overfitted the data, which is plausible given the large model sizes and the limited size of the FLIR dataset in particular. Consequently, despite their strong reported performance, these models are unlikely to generalize well to unseen data.

Next followed architecture optimization with Optuna, where we aimed to find smaller versions of our previously selected transformer architectures. All models had different sets of hyperparameters, and we restricted the search space to values that were smaller than or equal to those of the pretrained architectures. This allowed us to identify new architectures that significantly reduced both the number of model parameters and the number of FLOPs. For all models, we reduced the FLOPs to below 800 M, and for most models even further, to the range of 100–200 M, which is comparable to Axis's current classifier. This indicates that these transformer models could feasibly run on an embedded system.

After suitable architectures had been identified for each selected model, we trained them either from scratch using a cropped grayscale version of ImageNet-1k or using knowledge distillation. We began with an initial experiment comparing grayscale and RGB inputs, using CrossViT as a test case. The results showed that grayscale generally achieved higher accuracy, and we therefore trained all subsequent models using only grayscale images. We also observed early on that CrossViT did not perform sufficiently well in its reduced form and decided not to pursue further experiments with it. Instead, we used the larger CrossViT model as a teacher for distillation.

For the final evaluation, we included the train split of Axis' small dataset in the finetuning of the models that were trained from scratch along with the best performing distilled model, MaxViT with a CrossViT teacher. The distilled MaxViT model obtained the best score for recall of the person class and overall accuracy, but it came at a cost of too many false alarms, since the precision score was lower than that of Axis' original classifier. The best precision was obtained by Swin, but all other metrics were lower than the original classifier's. DeiT had the best recall animal score, but precision was slightly lower than that of the original classifier.

This study has produced many different results on several metrics. Evidently, some trans-

formers perform extraordinarily well on some metrics but considerably worse on others. The main goal for Axis is to find a transformer model that is better than the original classifier when all metrics are considered. When further examining the results from tables 5.15, 5.16 and 5.17, we see that such a transformer does indeed exist. MaxViT were able to outperform Axis' original convolution-based classifier on all metrics. It has fewer FLOPs and only about 0.5 million parameters, which is very lightweight. Table 5.18 summarizes the results of MaxViT from section 5.3.6, comparing them with the original classifier. To answer the first research question, yes, it is possible for a transformer based model to outperform Axis' original convolution-based classifier, even when scaled down to similar size.

However, for classification of small thermal images in general, one could argue that other models are better. This is because the models showed different performances across different metrics. Therefore, what is considered the best model will depend on which metrics you value the most. Figure 5.4 visualizes the overall accuracy obtained in the final evaluation and relates it to number of FLOPs and model size. Here we see that both the distilled version of MaxViT and DeiT achieved a higher overall accuracy than the current classifier. A similar analysis can be deducted on all the other metrics, generating different results. In this report, we deem MaxViT to be the best model for Axis' purposes, despite there being other models that perform better on individual metrics. The reason for this is that Axis value all metrics equally and do not wish to sacrifice any metric over another.

Comparing the two training approaches, distillation produced higher person-class recall, while training from scratch yielded more stable performance across all metrics. This is evident in the final evaluation, where the best-performing distilled model, MaxViT distilled with CrossViT, was outperformed by Axis' original classifier on recall of the animal class, even though it had the best results on other metrics. Finetuned MaxViT managed to be better than the original classifier on all considered metrics, causing no unwanted trade-off. Therefore, a finetuned model is a better alternative for Axis. However, finetuning does not have to be the best training approach in general.

When considering the results from training with distillation, we can see that it is a very effective training approach. Especially for MaxViT, which achieved the best results on two metrics. When finetuning, DeiT and MaxViT were the most successful models, generating the best results on two out of three metrics. To answer the second research question, distillation is the better training approach for classification when the model has a size constraint. This is evident because for the most part, distillation generated higher accuracies than finetuning. The reason for this is because it allows for more advanced pattern recognition and introduces more complexity into the model, without having to increase the model size.

Table 5.18: The table shows our best model, MaxViT, compared with the original classifier on the final evaluation in section 5.3.6.

Performance	MaxViT	Original classifier
Recall person class	**98.0**	97.9
Precision person class	**97.18**	96.9
Recall animal class	**97.1**	96.5
Overall accuracy	**96.31**	95.15
FLOPs (M)	**136**	143

Possible reasons for MaxViT's success could be its hybrid architecture. It leverages both convolutions, injecting spacial inductive bias, and powerful Multi-Axis attention which allows for reduced inference time. A way to reason about this is to compare it with Swin. Swin has a very similar architecture, because just like in MaxViT, the image is divided into a grid and attention is performed only between pixels that have the same relative position. However, Swin is a pure transformer and does not include any convolutions. MaxViT generally performs much better than Swin when both models are scaled down. This strengthens the argument that the hybrid architecture is a key contributor to MaxViT's success, because the models are otherwise so similar. It also highlights the importance of convolutions when needing a model that is both high-performing and lightweight.

Additionally, Maurício et al. (2023) highlighted in the report *Comparing Vision Transformers and Convolutional Neural Networks for Image Classification: A Literature Review* that vision transformers generally outperform CNNs when domain-specific training data is limited, which is the case for us. This is primarily because the attention mechanism enables transformers to learn more global and expressive patterns, even when the amount of domain-specific data is scarce. Although Axis's dataset contains approximately 480,000 training images, which is much less than what transformer models are usually trained on. For context, compare with the ImageNet-1k dataset that all of our pretrained transformers were trained on. It contains 1,2 million images for training. Additionally, Axis' large dataset consists of both thermal and RGB images, so the number of domain-specific images are even less. This is due to the fact that thermal images inherently are more difficult to obtain than RGB images, for several reasons. First, thermal cameras are significantly more expensive and less widely available. Second, thermal data is still challenging to synthesize realistically, even with state-of-the-art generative models such as GANs. Finally, there is a limited availability of large, open-source thermal datasets. As a result, research in the thermal domain often involves working with relatively small domain-specific datasets, which may further motivate the success of the transformers in this study.

One reason why Axis's original classifier remained difficult to outperform, particularly across multiple evaluation metrics, is that it was trained exclusively on Axis's proprietary datasets. In contrast, the transformer models were either pretrained on ImageNet-1k or trained using distillation, and only subsequently finetuned on Axis's large dataset. Transformers generally require substantially more pretraining data than CNNs to be successful, which is why we could not train them solely on Axis's data. However, for niche classification domains where data is scarce, like for classification of thermal images, transformers generally perform better than CNNs (Maurício et al., 2023). This is because after pretraining, transformers need less domain specific finetuning data than CNNs to be successful. So overall, transformers require more data, but only for the pretraining. In the thermal domain, transformers show better results than CNNs because they are better at pattern recognition even when the finetuning dataset is small.

As a consequence of needing more vast pretraining, the transformer models were exposed to a much greater diversity of training data, which does improve their ability to generalize to unseen samples. However, this broader data distribution can also negatively impact performance when the training data differs significantly from the target domain, as is the case for thermal imagery. This is because although converted to grayscale, the ImageNet-1k dataset consists entirely of RGB images. It is likely that models trained exclusively on domain-specific data may achieve better performance in that domain. However, because of

the limited amount of labeled thermal image data, the transformers had to endure this disadvantage.

Ultimately, our goal is to develop a model trained on as much relevant data as possible while maintaining strong domain alignment. For this reason, we selected the best-performing model for Axis' use case from those further finetuned on Axis's smaller, domain-specific dataset. We also saw that these models were boosted in their ability to classify animals correctly, since that dataset mainly contains animals. This is evident when comparing the results in tables 5.10 and 5.13 with the results in table 5.16, where further finetuning on animals improved the recall of the animal class for all models. A better ability to correctly classify animals correlates with confusing animals for humans less times, which reduces false alarms.

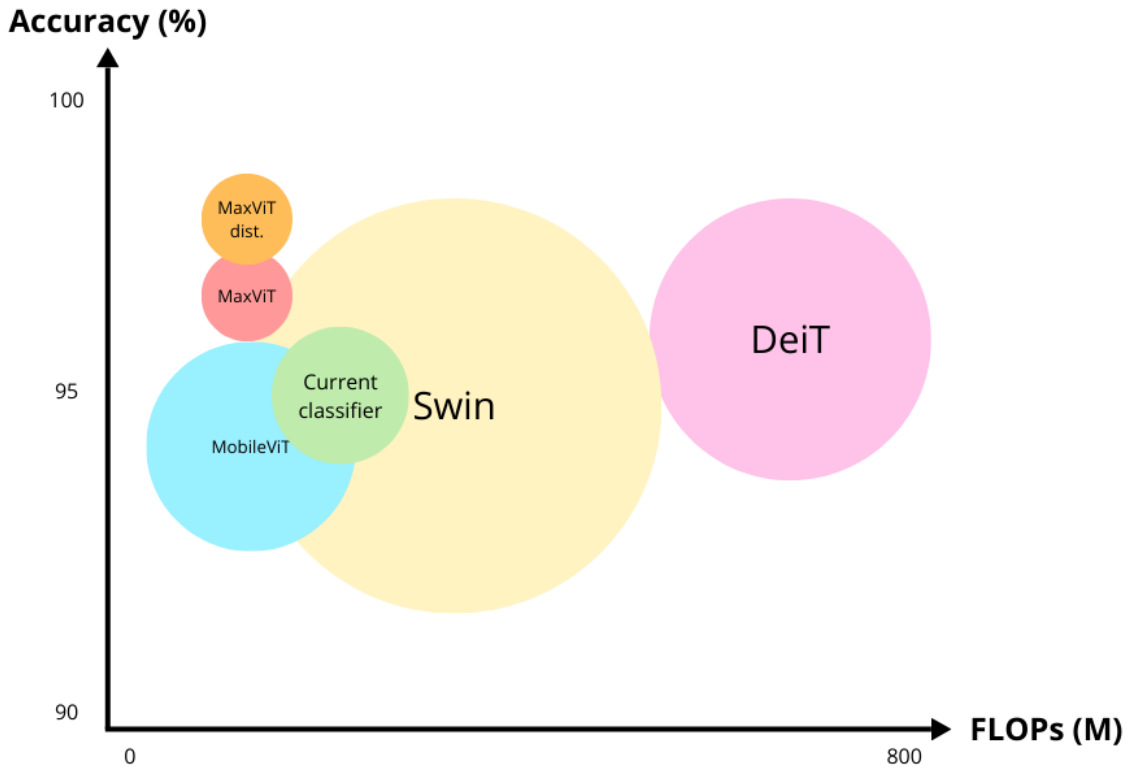


Figure 5.4: Here is a visualization of the results from section 5.3.6, the final overall accuracy after finetuning on Axis' small dataset of animals. Each circle represents a classifier. The name of the classifier can be seen in the middle of the circle. On the y-axis, we have the overall accuracy, on the x-axis, we have the number of FLOPs. The size of the circles correlates with the number of model parameters. The center of the circle determines where to read the value on the axes. The bigger the circle, the more parameters. The color does not have any meaning, it is just to help distinguish the circles.

Are transformers a better choice for small thermal image classification in general? This is the last research question that we aim to answer. In this study, transformers generally outperformed all of our baseline CNN models, except for EfficientNet that reached a better result of the recall of the animal class after it had ben distilled with CrossViT. CNNs naturally have a lower number of parameters and FLOPs than transformers and they require less pretraining data. Consequently, without any optimization of parameters and architecture, CNNs are the best choice. This is because all the transformer models are way to big initially, which

does not adhere to our size and inference time constraints. However, with optimization and and down-scaling, transformers are the better choice. This is because the attention mechanism is better at finding complex patterns than local convolution filters. It is possible that because of the low resolution and lack of colors, textures and finer details in thermal images, a global understanding of the image is of greater importance for the model's success, making transformers superior. However, our best performing transformer, MaxViT, is a hybrid transformer because it includes convolutions in addition to having attention. This shows the importance of convolutions, especially to be able to reduce model size and shorten inference time.

5.4.2 Future work

Although our study of transformers for classification of thermal images was extensive, there remain many opportunities for further experimentation and improvement. First, additional transformers could have been included. For example, the ACC-ViT model proposed by Ibtehaz et al. (2024) as a "strong vision backbone, which is also competitive in mobile-scale versions, ideal for niche applications with small datasets" which would have been suitable for our use case. Another interesting transformer is EfficientViT by Liu et al. (2023), which, while similar to the EfficientFormer, adopts a different approach to achieving efficiency on mobile devices.

Second, further exploration of model architectures and training hyperparameters could be conducted. The training settings used in this study were not necessarily optimal for all models, and it is likely that alternative architectural choices or hyperparameter configurations could yield improved performance. However, due to limitations in time and computational resources, we were unable to conduct a more exhaustive search. Since we identified MaxViT as a model that outperformed the original classifier across all evaluation metrics, future work could focus on more thoroughly optimizing this architecture.

Finally, additional experiments with knowledge distillation could be performed. The distillation process could be extended by incorporating intermediate feature-level distillation from earlier layers of the teacher model, as well as by experimenting with alternative teacher architectures. For instance, the Swin model would probably be a suitable teacher model because of its large size and complex shifted windows design. These directions may further improve student model performance and generalization.

Chapter 6
Conclusions

This work investigated whether modern vision transformer architectures can outperform convolutional neural networks for thermal image classification under realistic constraints on model size and computational cost. Through extensive experimentation across several datasets, multiple architectures, and two training strategies, we demonstrated that transformer-based models are not only competitive with, but can surpass, a carefully optimized CNN baseline for this use case.

Among all evaluated models, we deemed MaxViT to be the most successful, surpassing the original CNN model on all metrics in the final evaluation. It achieved higher recall and precision for the person class, higher recall for the animal class, and higher overall accuracy. Importantly, this improvement was obtained with a model containing only approximately half a million parameters and 136 M FLOPs, making it suitable for deployment in embedded systems.

The results further suggest that hybrid transformer architectures, which combine convolutional inductive biases with global attention mechanisms, are particularly well-suited for thermal image classification, since training data is limited and contains images of low resolution. Knowledge distillation proved to be a powerful tool for boosting recall, although it introduced a trade-off with precision in some cases. Training models from scratch produced more stable results across all metrics.

While the original CNN classifier remained competitive on individual metrics, this study shows that carefully designed and optimized transformers can provide a better overall trade-off between performance and efficiency. Consequently, MaxViT represents a viable and promising replacement for Axis's current classifier, and more broadly, this work supports the use of transformer-based models for real-world thermal vision applications.

References

Agrawal, D. and Karar, V. (2018). Color palette selection in thermal imaging for enhancing situation awareness during detection-recognition tasks. In *2018 International Conference on Recent Innovations in Electrical, Electronics & Communication Engineering (ICRIEECE)*, pages 1227–1232.

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A Next-Generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2623–2631.

Bhojanapalli, S., Chakrabarti, A., Glasner, D., Li, D., Unterthiner, T., and Veit, A. (2021). Understanding robustness of transformers for image classification.

Chen, C.-F., Fan, Q., and Panda, R. (2021). Crossvit: Cross-attention multi-scale vision transformer for image classification.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale.

Ibtehaz, N., Yan, N., Mortazavi, M., and Kihara, D. (2024). Acc-vit : Atrous convolution's comeback in vision transformers.

Imaging, T. (2024). Flir data set dataset. `https://universe.roboflow.com/thermal-imaging-0hwfw/flir-data-set`. visited on 2025-10-29.

Li, Y., Hu, J., Wen, Y., Evangelidis, G., Salahi, K., Wang, Y., Tulyakov, S., and Ren, J. (2023). Rethinking vision transformers for mobilenet size and speed.

Liu, X., Peng, H., Zheng, N., Yang, Y., Hu, H., and Yuan, Y. (2023). Efficientvit: Memory efficient vision transformer with cascaded group attention.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows.

Maurício, J., Domingues, I., and Bernardino, J. (2023). Comparing vision transformers and convolutional neural networks for image classification: A literature review. *Applied Sciences*, 13(9).

Mehta, S. and Rastegari, M. (2022). Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer.

Nguyen, T. X. B., Rosser, K., and Chahl, J. (2021). A review of modern thermal imaging sensor technology and applications for autonomous aerial navigation. *Journal of Imaging*, 7(10).

Nunnari, G. and Calvari, S. (2025). Exploring vision transformers and convolution neural networks for the thermal image classification of volcanic activity. *Applied Sciences*, 15(5).

O'Shea, K. and Nash, R. (2015). An introduction to convolutional neural networks.

Qazi, T., Lall, B., and Mukherjee, P. (2025). Thermaldiff: A diffusion architecture for thermal image synthesis. *Journal of Visual Communication and Image Representation*, 111:104524.

Tan, M. and Le, Q. V. (2020). Efficientnet: Rethinking model scaling for convolutional neural networks.

Tan, M. and Le, Q. V. (2021). Efficientnetv2: Smaller models and faster training.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. (2021). Training data-efficient image transformers & distillation through attention.

Tu, Z., Talebi, H., Zhang, H., Yang, F., Milanfar, P., Bovik, A., and Li, Y. (2022). Maxvit: Multi-axis vision transformer.

Ulusoy, U., Yılmaz, K., and Özşahin, G. (2022). Generative adversarial network for generating synthetic infrared image from visible image. *Gazi Üniversitesi Fen Bilimleri Dergisi Part C: Tasarım Ve Teknoloji*, 10(2):286–299.

Vasu, P. K. A., Gabriel, J., Zhu, J., Tuzel, O., and Ranjan, A. (2023). Fastvit: A fast hybrid vision transformer using structural reparameterization.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need.

Wang, W., Xie, E., Li, X., Fan, D.-P., Song, K., Liang, D., Lu, T., Luo, P., and Shao, L. (2021). Pyramid vision transformer: A versatile backbone for dense prediction without convolutions.

Wang, W., Zhang, J., and Shen, C. (2010). Improved human detection and classification in thermal images. In *2010 IEEE International Conference on Image Processing*, pages 2313–2316.

Wightman, R., Touvron, H., and Jégou, H. (2021). Resnet strikes back: An improved training procedure in timm.

Wikipedia contributors (2026a). Transformer (deep learning) — Wikipedia, the free encyclopedia. [Online; accessed 3-February-2026].

Wikipedia contributors (2026b). Vision transformer — Wikipedia, the free encyclopedia. [Online; accessed 3-February-2026].

Xu, C., Gao, W., Li, T., et al. (2023). Teacher–student collaborative knowledge distillation for image classification. *Applied Intelligence*, 53.

.1 Appendix: confusion matrices

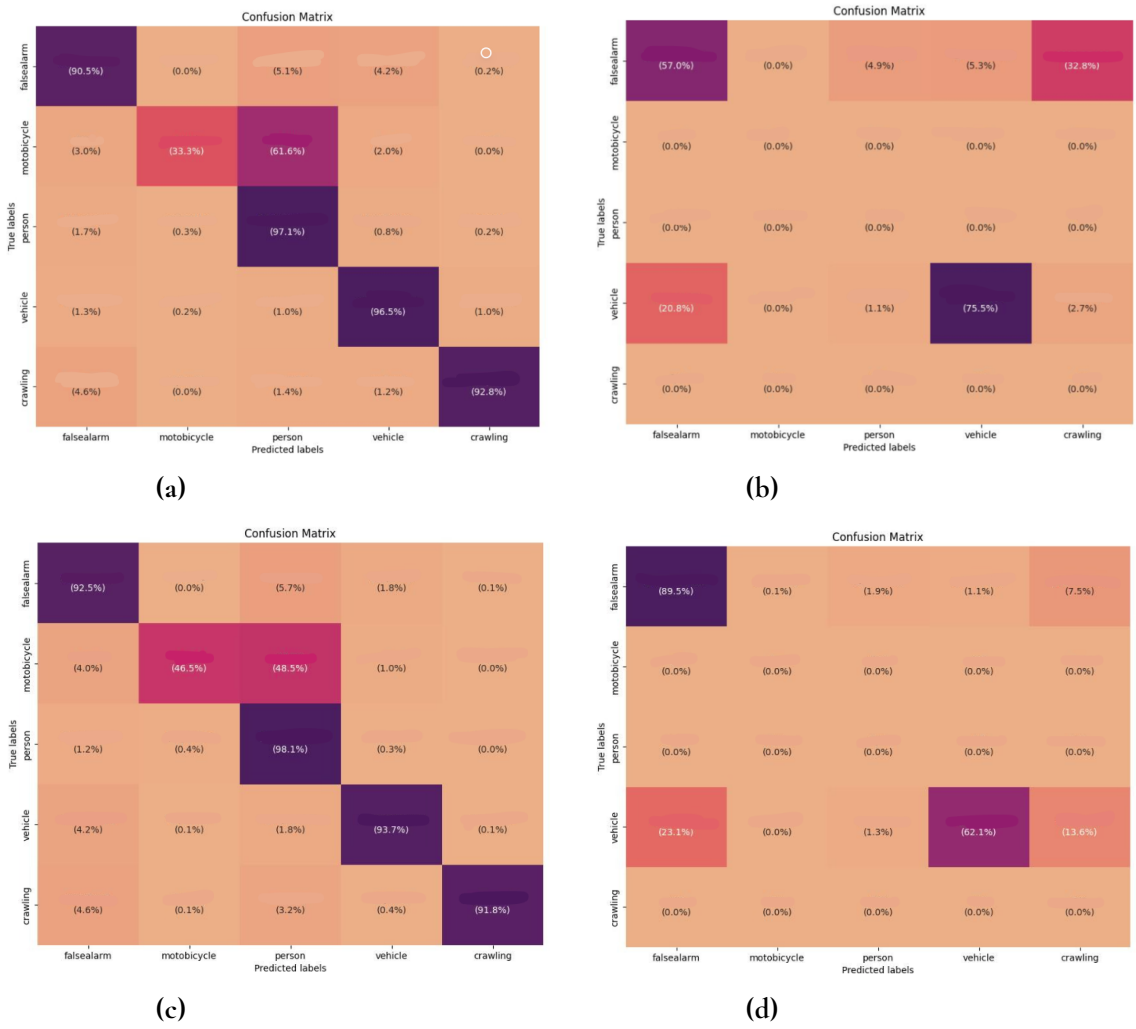


Figure 1: Confusion matrices from section 5.3.4.

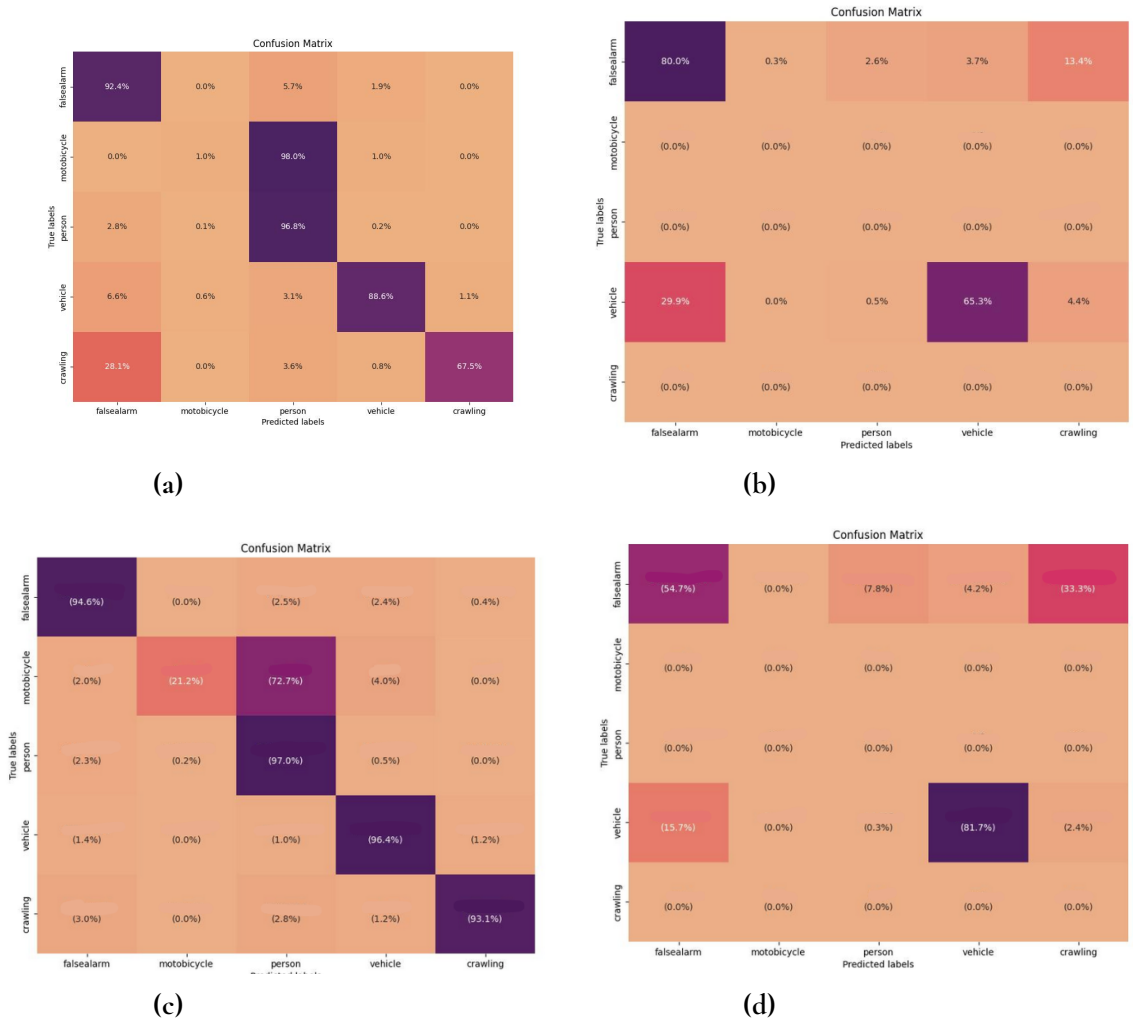


Figure 2: Confusion matrices from section 5.3.4.

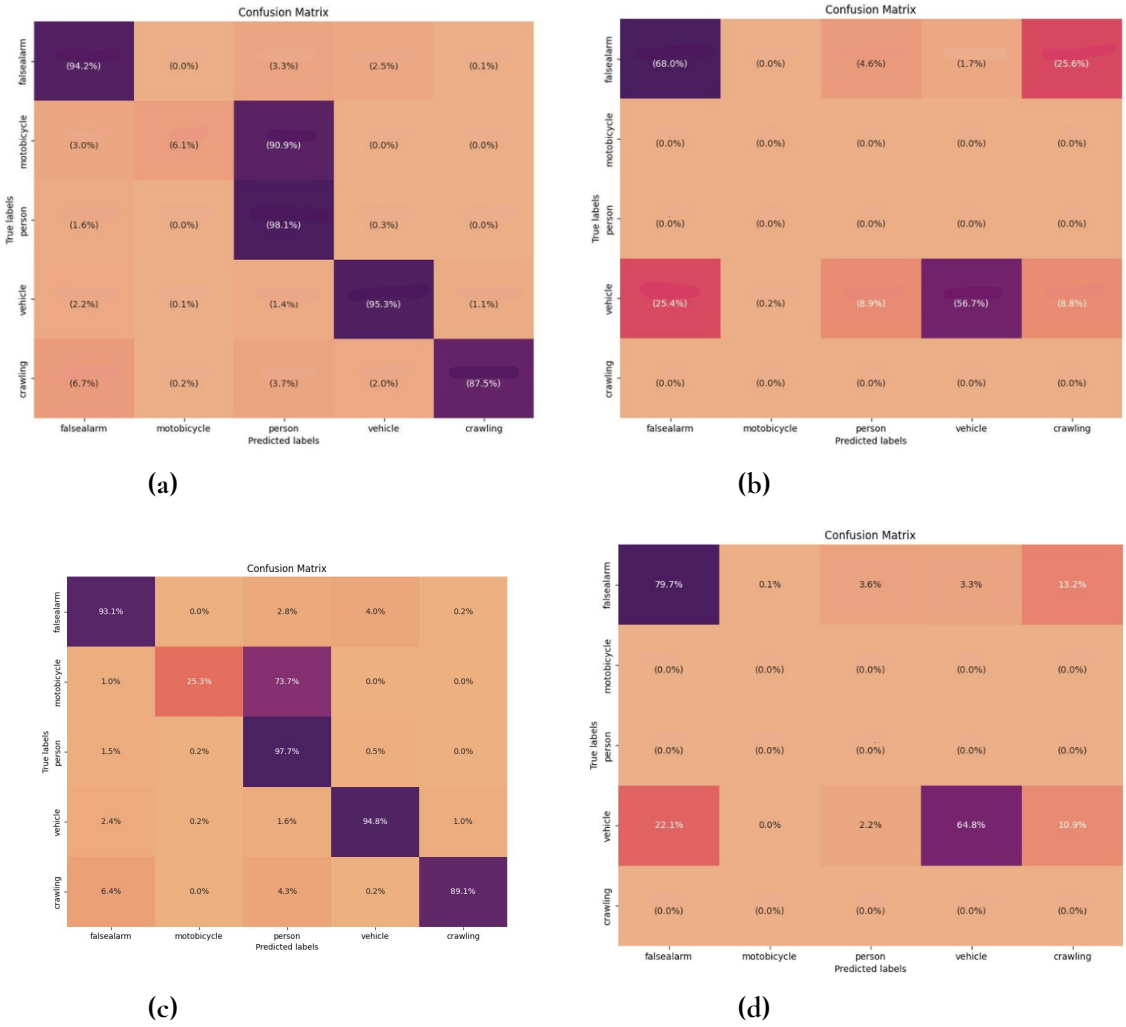


Figure 3: Confusion matrices from section 5.3.4.

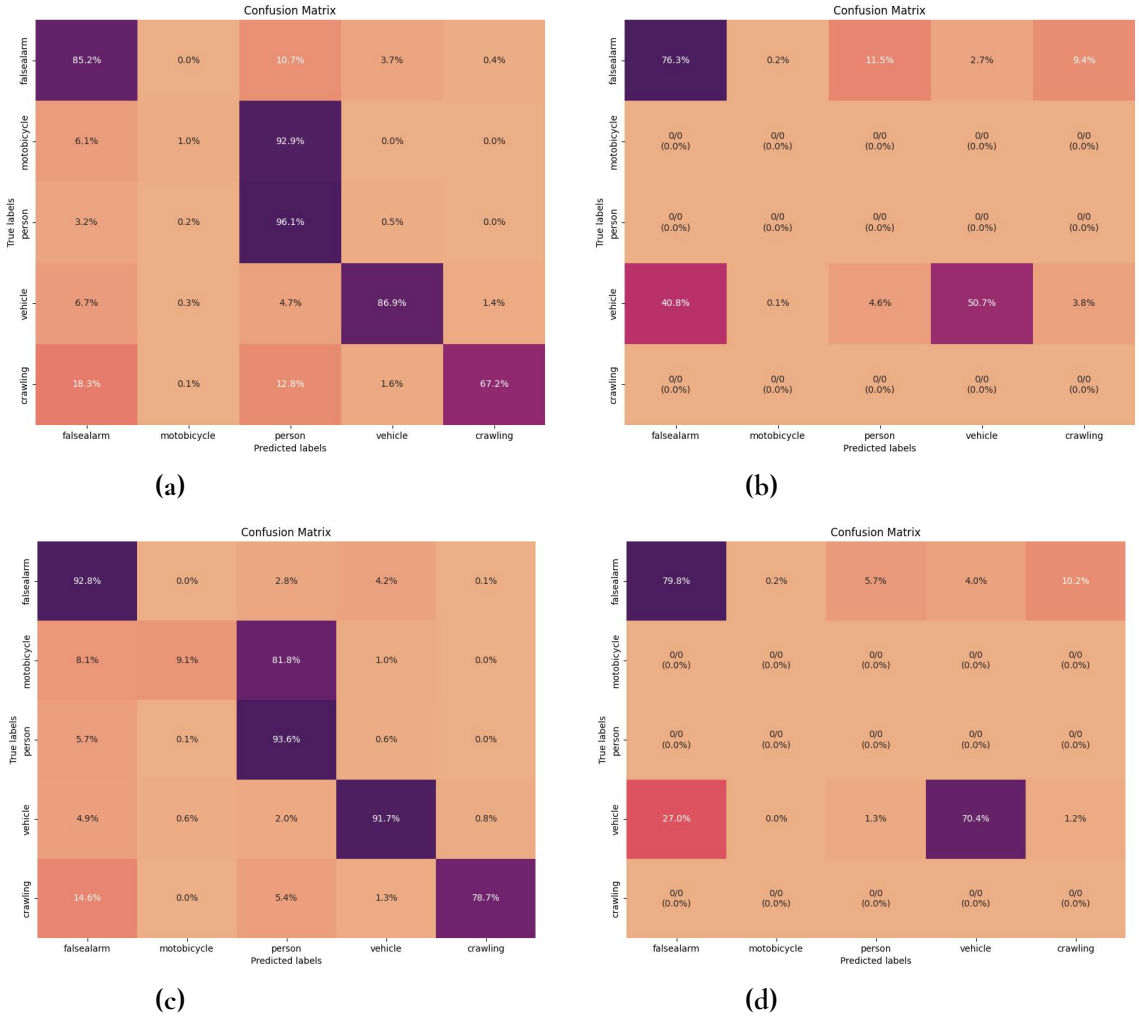


Figure 4: Confusion matrices from section 5.3.4.

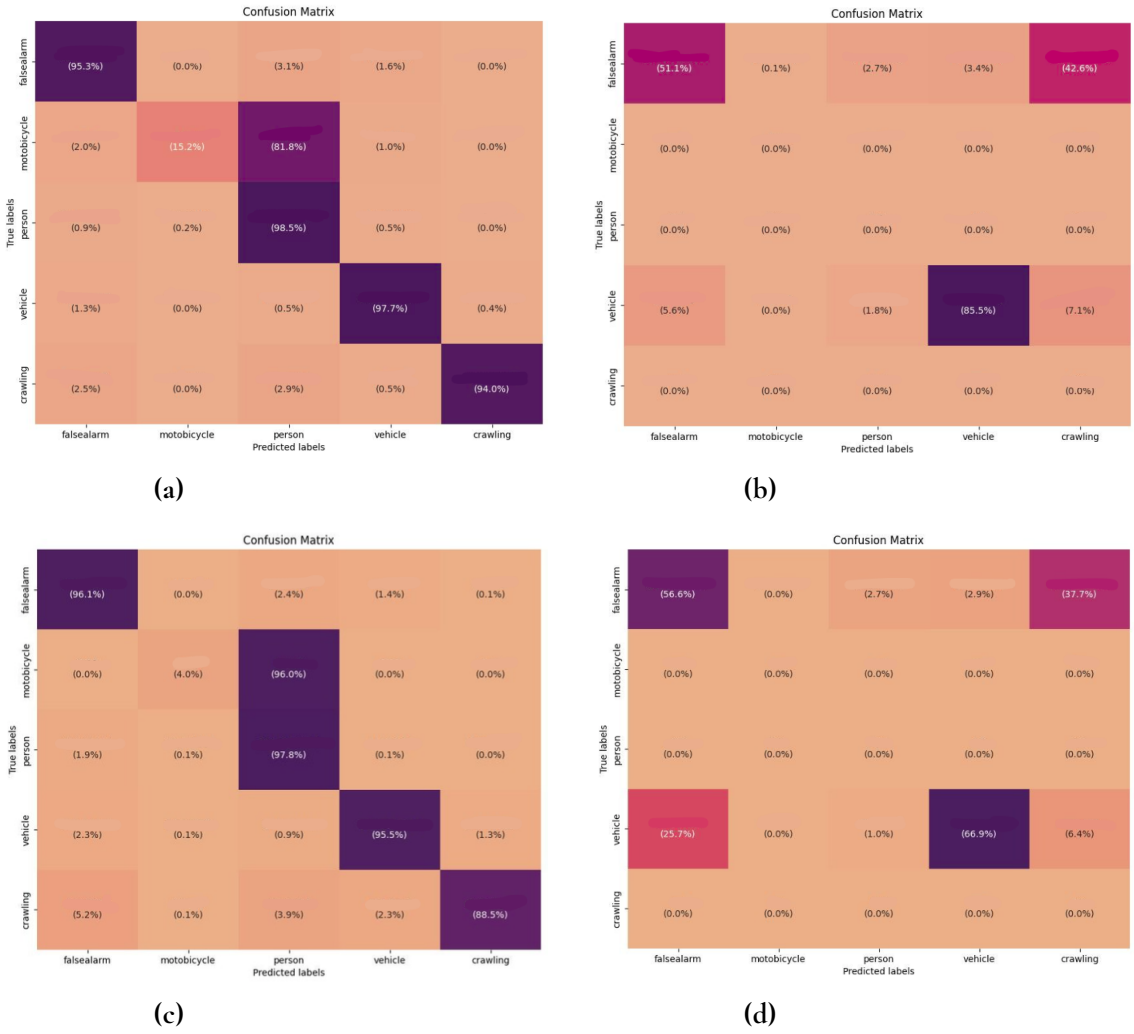


Figure 5: Confusion matrices from section 5.3.5.

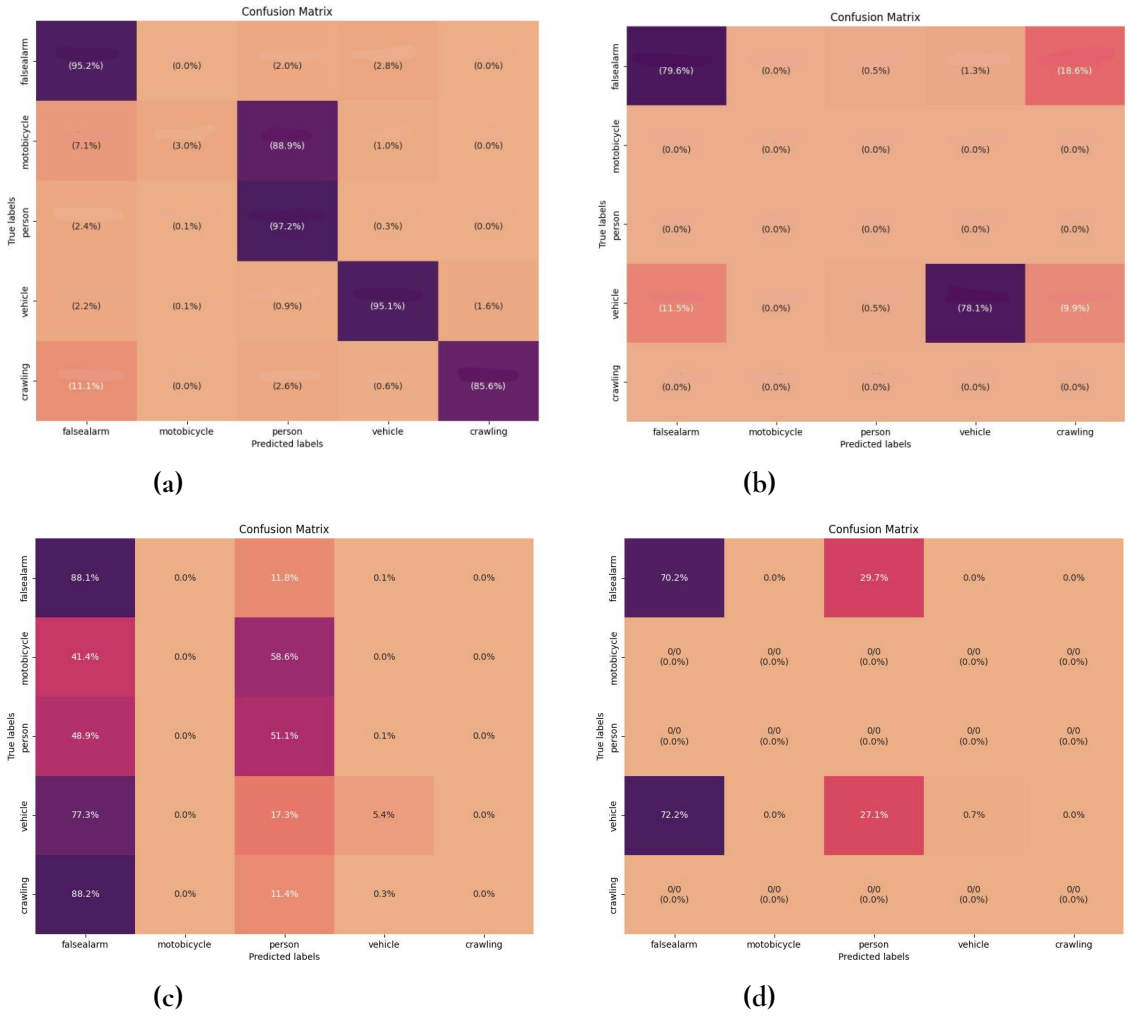


Figure 6: Confusion matrices from section 5.3.5.

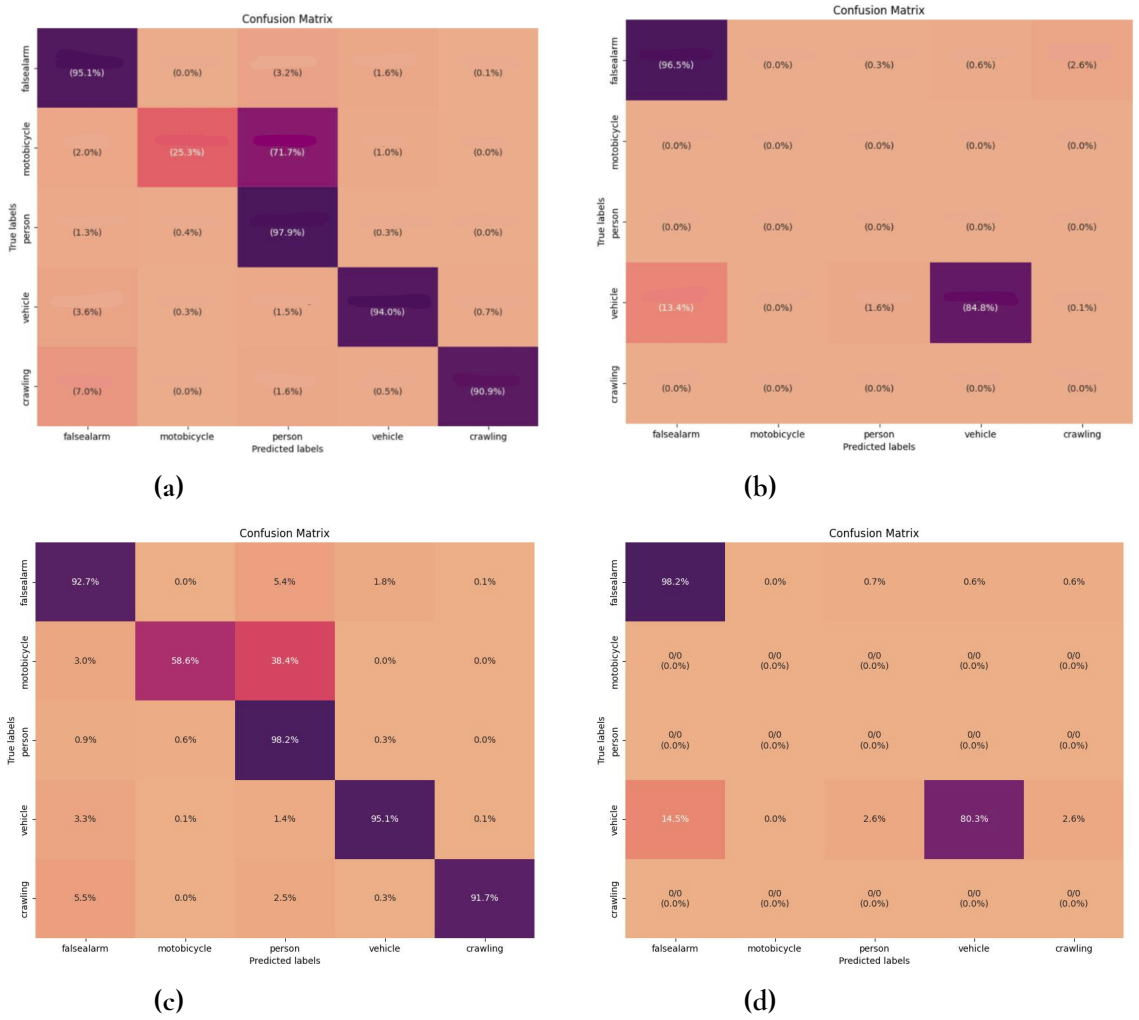


Figure 7: Confusion matrices from section 5.3.6.

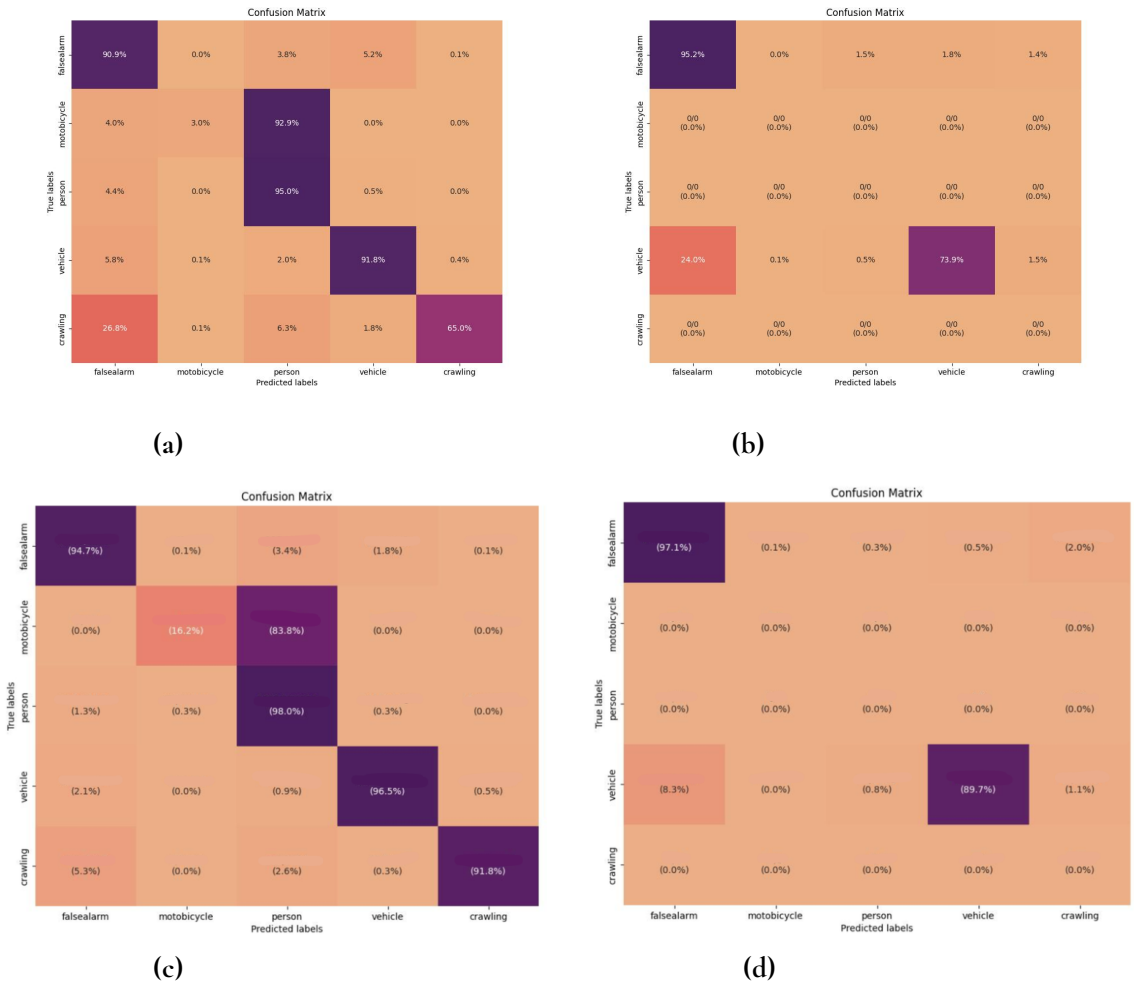


Figure 8: Confusion matrices from section 5.3.6.

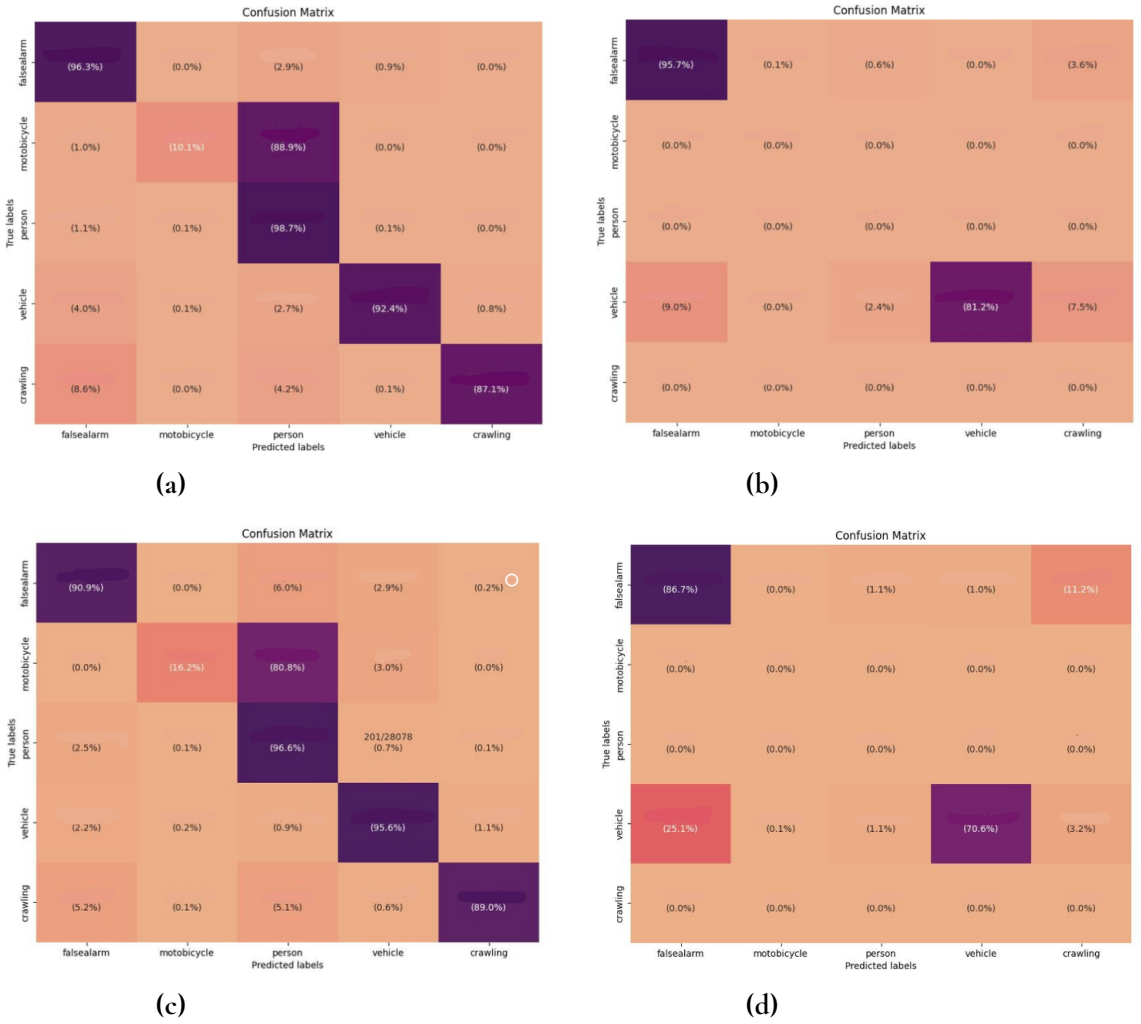
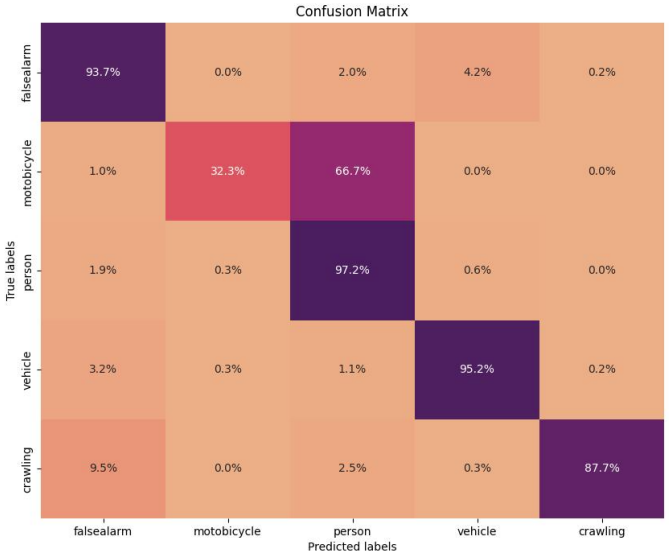
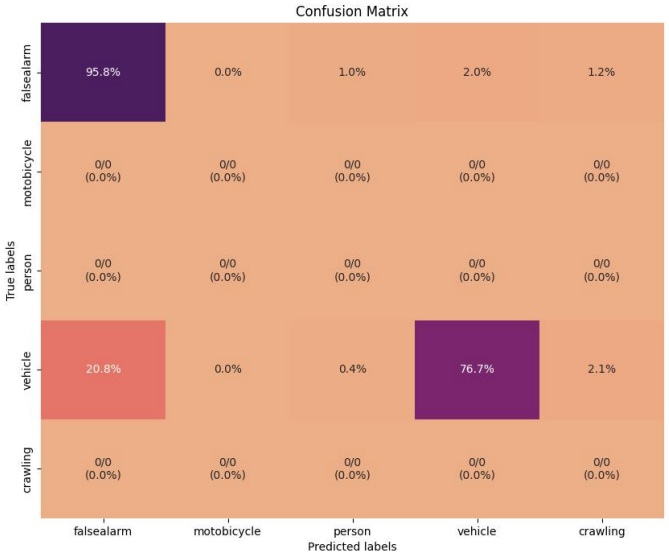


Figure 9: Confusion matrices from section 5.3.6.



(a)



(b)

Figure 10: Confusion matrices from section 5.3.6.

EXAMENSARBETE Classification of Small Thermal Images with Transformers
An Evaluation for Security Applications
STUDENT Elin Persson
HANDLEDARE Pierre Nugues (LTH)
EXAMINATOR Eren Aksoy (LTH)

Små bilder, stora beslut

POPULÄRVETENSKAPLIG SAMMANFATTNING **Elin Persson**

Säkerhetssystem använder allt oftare AI för att avgöra om en människa finns i bild. Detta arbete undersöker hur transformerbaserade modeller kan förbättra klassificeringen av små termiska bilder under begränsande resurskrav.

Ett modernt säkerhetssystem ska kunna avgöra om det är en människa i bild och larma därefter. Ibland kan det dock vara helt andra saker som utlöser larmen, speciellt när värmekamror används. Detta eftersom de har falska färger och sämre upplösning. Exempelvis är det vanligt att djur och människor förväxlas, vilket orsakar falsklarm för säkerhetsystem. Falsklarm är kostsamma, irriterande och gör att användare tappar förtroende för systemet.

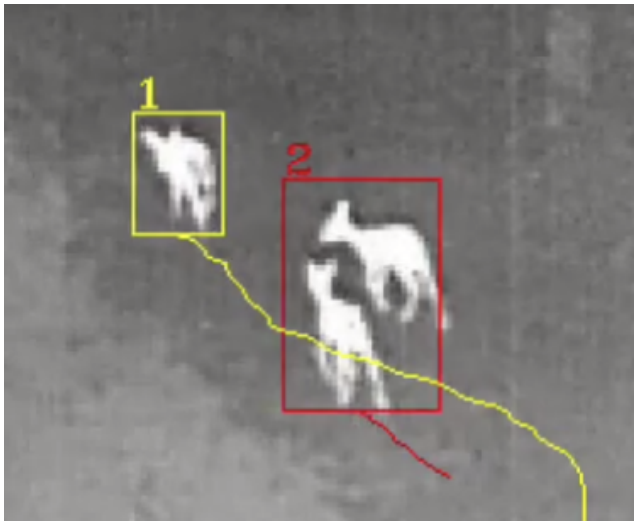


Figure 1: Här ser vi tre rådjur. De två i den röda rutan har detekterats och klassificerats som en människa av AI-modellen. Detta orsakar att säkerhetssystemet producerar ett falsklarm.

I mitt examensarbete har jag undersökt flera varianter av en bildigenkännande AI-modell som kallas transformer. Jag har jämfört den med en annan sorts AI-modell, convolutional neural network (CNN), som används i större utsträckning vid bildigenkänning. De två AI-typerna har fundamentala arkitekturskillnader. Fokus ligger på termiska kameror, där bilderna är mycket små, gråskaliga och ofta saknar tydliga detaljer. Klassificering av termiska bilder är mindre utforskat än normala bilder, och mängden tillgänglig data är liten. Det gör uppgiften särskilt svår även för avancerade AI-modeller.

Resultatet visar att transformer-modeller kan prestera likvärdigt eller bättre än CNN-modeller, även under pressade resurskrav. Den bästa transformer-modellen lyckades minska antalet falsklarm jämfört med den modell som används i systemet idag. Samtidigt var transformern både snabbare och betydligt mindre än den tidigare modellen. Det innebär att systemet blir mer pålitligt och mindre kostsamt.

Arbetet visar att det är möjligt att bygga smartare säkerhetssystem som fattar bättre beslut, även när informationen är begränsad. På sikt kan detta leda till färre onödiga utryckningar, lägre kostnader och tryggare övervakning.