UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Data Science Curriculum

Karl Kaspar Haavel

Exploring Out-of-Distribution Detection Using Vision Transformers

Master's Thesis (15 ECTS)

Supervisor(s): Meelis Kull, PhD
Bhawani Shankar Leelar, PhD

Tartu 2022

Exploring Out-of-Distribution Detection Using Vision Transformers

Abstract:

Current state-of-the-art artificial neural network (ANN) image classifiers perform well on input data from the same distribution that it was trained with, also known as in-distribution (InD), yet have worse results on out-of-distribution (OOD) samples. An input can be considered OOD for many reasons - such as an input with a new concept (e.g. new class), or the input has random noise generated by a sensor. Knowing if a new data point is OOD is necessary for deploying models in real-world safety-critical applications (e.g. self-driving cars, healthcare) to make safer decisions. For example, a self-driving car slows down when it detects an OOD object or gives the control back to the human. The primary method used for OOD detection is to utilise ANN as a feature extractor of embeddings to approximate where the new data point will be in the embedding space and compare it to trained embeddings using distance metrics. We use a Vision Transformer (ViT) as the ANN because there has been a push to use large-scale pre-trained Transformers to improve a range of OOD tasks. Improvements stem from ViT's state-of-the-art performance as a feature extractor, which can be used out-of-the-box for OOD detection compared to convolutional neural networks (CNNs), which require custom training methods and exposure to OOD to reach similar results.

In this thesis, a ViT was used as a feature extractor, and the performance of OOD detection was compared using various distance metrics to determine the robustness and choose the best distance metric in ViT's embedding space. Three separate experiments were conducted with multiple datasets, methods, models and approaches. The experiments showed that ViT is capable of OOD detection out-of-the-box without any custom training methods or exposure to OOD. However, none of the distance metrics could noticeably improve the results of OOD detection obtained with the baseline Mahalanobis distance. Nonetheless, ViT has considerably better OOD detection performance in most datasets and is more generalisable than a similarly trained CNN. Furthermore, ViT is more robust to various distance metrics, proving that the features extracted from the model are good enough to discriminate between InD and OOD. Finally, it was shown that ViT with Mahalanobis distance has the best OOD detection performance when blending InD and OOD at various ratios. Future work can consider ensembling multiple distance metrics to utilise the properties of each distance metric and to apply the same methodology on other ANN architectures.

Keywords:

deep learning, neural networks, vision transformer, out-of-distribution detection

CERCS:

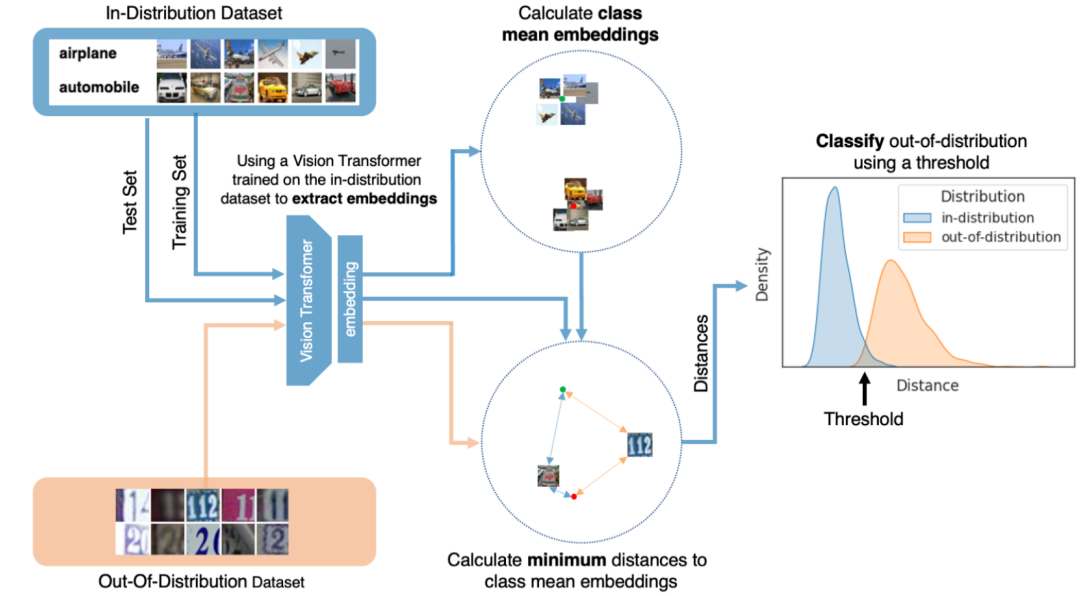
T111 - Imaging, image processing; P176 - Artificial intelligence

Visual Abstract:

Exploring Out-of-Distribution Detection Using Vision Transformers

Author: Karl Kaspar Haavel

Supervisors: Meelis Kull (PhD), Bhawani Shankar Leelar (PhD)



Jaotusvälisuse tuvastus nägemistransformeritega

Lühikokkuvõte:

Hetke parimad tehisnärvivõrkudel baseeruvad pildi klassifitseerijad töötavad hästi andmetel, mis pärinevad samasugusest jaotusest, millel neid treeniti. Probleeme esineb jaotusväliste sisenditega. Sisend võib olla jaotusväline mitmel põhjusel, nagu näiteks, sisend sisaldab uut tüüpi objekti (e.g. uus klass) või sisendil on juhuslik sensorimüra. Sisendi klassifitseerimine jaotusväliseks on tähtis, kui mudelid võetakse kasutusele ohutuse seisukorrast kriitilises keskkonnas (e.g. tervisevaldkond ja isejuhtivad autod). Ohutuse seisukorrast kriitilises keskonnas on vale otsuse tegemine kulukas, kuid kui mudel on teadlik jaotusvälisest sisendist, on võimalik vastava ohutu otsuse tegemise anda edasi inimesele. Jaotusväliste sisendite tuvamstamiseks, kasutatakse mudeli hõlmanguid. Mudelit kasutatakse hõlmangute ekstraheerijana, et saada teada uue sisendi ligikaudne asukoht hõlmanguruumis ning võrrelda mudelit treenitud sisendite hõlmangutega kasutades kaugusmõõdikuid. Hetketrend on kasutada suuremahulise eeltreeningu läbinud Transformereid, et parendada jaotusvälise tuvastuse tulemust, eriti Nägemistransformereid (ViT). ViT'dega on saavutatud tipptasemel tulemused jaotusvälise tuvastuses ilma kohandatud treenimismeetotite või kokkupuudet jaotusväliste sisenditega võrreldes konvolutsiooniliste närvivõrkudega (CNN).

Antud lõputöös kasutame ViT'd hõlmangute ekstraheerijana ja võrdleme jaotusvälise tuvastuse tulemusi kasutades valitud kaugusmõõdikuid, nagu näiteks eukleidiline, koosinus, standardiseeritud eukleidiline. Tegemaks kindlaks ViT robustsust ja määrata, mis on parim kaugusmõõdik ViT hõlmanguruumis, viime läbi kolm eksperimenti mitme andmekogumi, meetodi, mudeli ja lähenemisega. Eksperimentide tulemused näitasid, et ViT on võimeline jaotusvälise tuvastusega tegelema ilma kohandatud treenimismeetodite ja jaotusväliste sisenditega. Kuid ükski kaugusmõõdik ei suutnud märkimisväärselt parendada tulemust, mis saavutati baasmeetodit kasutades Mahalanobise kaugust. Sellest hoolimata, ViT jaotusvälise tuvastuse tulemused olid märkimisväärselt paremad enamus andmekogumites ja üldistavamad võrreldes sarnaselt treenitud CNN'iga. Lisaks sellele, ViT on robustne ja saavutab sarnased tulemused kasutades erinevaid kaugusmõõdikuid. See tähendab, et ViT manused on suutelised diskrimineerima sise- ja välijaotuste vahel. Lõpetuseks, näitame, et ViT'l on parem jaotusvälisuse tuvastus võime kui segatakse omavahel sise- ja välijaotuse andmekogumite pildid. Tulevikus võib kaaluda, mitme kaugusmõõdiku koosmõju, et kasutada ära erinevate kaugusmõõdiku omadused ja metotoloogia rakendamine teiste tehisnärvivõrkude arhitektuuridele.

Võtmesõnad:

süvaõpe, tehisnärvivõrgud, nägemistransformer, jaotusvälisuse tuvastus

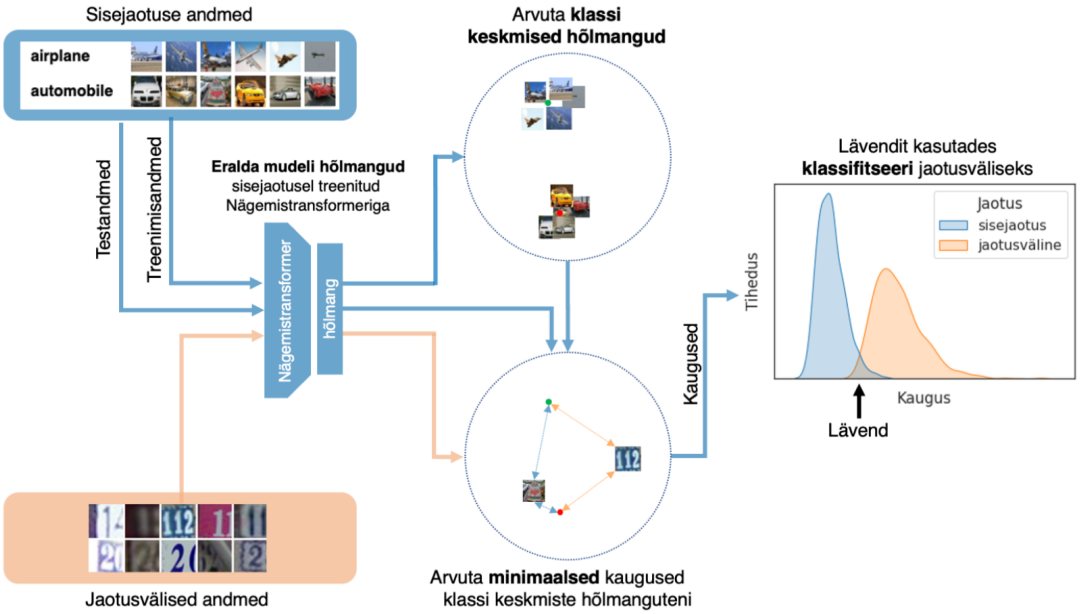
CERCS:

T111 - Pilditehnika; P176 - Tehisintellekt

Visuaalne kokkuvõte:

Jaotusvälisuse tuvastus nägemistransformeritega

Autor: Karl Kaspar Haavel
Juhendajad: Meelis Kull (PhD), Bhawani Shankar Leelar (PhD)



Contents

1 Introduction

Deep neural networks (DNNs) work well with data from the same distribution used for training and testing, known as in-distribution (InD). On the other hand, out-of-distribution (OOD) data can severely hamper the algorithm's performance. There are multiple reasons for the hampered performance of DNNs, such as misclassification inputs that are not in any of the training classes with high confidence [NYC15]. An input can be considered OOD due to many reasons, such as an input introducing a new concept (a new class) or the input having random noise that was generated by the sensor [Kon+21]. However, DNNs are used more and more in safety-critical applications, such as healthcare, autonomous driving and surveillance systems, leading to a need for models to distinguish InD from OOD [Amo+16], or in other words developing methods for reliable OOD detection. Reliable OOD detection can lead to safety-critical applications being aware of OOD inputs, reducing the risk of incorrect decisions by handing over the decision to act on the OOD input to humans or initiate fail-safe mechanisms (e.g. stopping the car).

The state-of-the-art (SOTA) deep learning methods in image classification and OOD detection were based on convolutional neural networks (CNNs), such as ResNet [He+16]. Meanwhile, Transformer [Vas+17] based architectures, such as BERT [Dev+18] have become the *de-facto* standard in natural language processing tasks, but their use in computer vision has been limited. This was true until the development of the Vision Transformer (ViT) architecture [Dos+20], which has shown to achieve SOTA results in image classification using a purely Transformer based architecture without any convolution. ViT's success, similarly to Transformer based architectures in other domains, relies on large-scale pre-training to achieve SOTA results and then fine-tuning for smaller tasks.

A common approach for OOD detection is to use the embeddings of any trained classifier to calculate class mean embeddings and use the distance from the class mean embedding to the inputs embedding as the OOD score [Lee+18]. Embeddings are the pre-logit layer of a trained softmax classifier. A good model embedding space (aka latent space) for OOD detection will have InD samples near each other, while an OOD sample should lie further away. Any distance metric can be used to calculate the distance between the class mean embedding and the input embedding, with the most popular and best performing for OOD detection being Mahalanobis distance. Until lately, to achieve SOTA OOD detection performance, the methods used contrastive loss [Win+20], or outlier exposure requiring training or fine-tuning with OOD data [Hen+19; FRL21]. Hence, making these methods harder to use out-of-the-box by needing access to OOD data. Koner et al. [Kon+21] overcame these shortcomings with their proposed OODformer and obtained SOTA results in various OOD detection benchmarks showing that pre-trained ViT, if fine-tuned on an InD dataset, achieve good class embeddings that can discriminate OOD samples. For these reasons, it was found worthwhile to extend the work of OODformer by applying various distance metrics to explore how OODformer

reacts to OOD detection. We first replicate the initial work of OODformer and then perform various experiments to observe its response. Moreover, implement a novel experimentation method called mixer images to explore ViT OOD detection performance to distribution shift from InD to OOD in a controlled manner.

1.1 Contributions

The contributions of this thesis are the following:

- We confirm the previous findings of OODformer that ViT's have good OOD detection out-of-the-box and do not need OOD data to improve OOD detection.
- We extend the work of OODformer [Kon+21] by investigating the potential of additional distance metrics not considered in the original work.
- We confirm that ViT embeddings used for OOD detection are robust to various distances.
- We explore OOD distance metrics with distribution shift from InD to OOD by using a novel methodology of mixing the information of InD and OOD images, which we call mixer images.
- We show that mixer images provide valuable insights into distribution shift.
- We show that a pre-trained ViT is more sensitive to distribution shift than a pre-trained ResNet architecture.

1.2 Outline

- **Background**: Gives a general overview of deep learning, CNN's, ViT's, and related works of OOD detection.
- **Methodology**: Provides a brief overview of the datasets, distance and performance metrics, defines the method for OOD detection and the mixer image experiments.
- **Experiments**: Describes the conducted experiments and the results of the experiments. A short description and discussion of the results follow each experiment.
- **Conclusion**: Concludes the thesis and lists the possible future avenues and limitations of this work.

2 Background

The background chapter introduces the essential concepts needed to understand the thesis. It gives a brief overview of feedforward neural networks, deep learning, convolutional neural networks, Transformers, Vision Transformers and ends with an overview of OOD detection.

2.1 Feed-Forward Neural Networks and Deep Learning

Feed-Forward Networks (FFNs), also called MultiLayer Perceptrons (MLPs) or Artificial Neural Networks (ANNs), are the building blocks of deep learning and the first type of artificial neural networks developed [Sch15]. The definition of deep learning is generally using machine learning algorithms based on neural networks [Bro20], which can be thought of as making algorithms learn by example. Due to the broad scope of the field of deep learning, an in-depth examination of its mechanics is outside the scope of this thesis. Hence, only the necessary details will be covered, which will be needed for further discussions in later chapters.

FFNs are known to be universal function approximators [Zha+21] due to being able to learn any nonlinear function f and can map any input x to the output y. The models are called *feedforward* due to the information flowing through the network without any feedback connections where the output of the model is fed back into itself. If the network is extended to include feedback connections, it is called a recurrent neural network (RNN). FFNs are *networks* because they are typically composed together of many different functions and depicted as a directed acyclic graph describing how the functions are linked with one another. FFN can consist of any number of functions f^n that are connected with each other in a chain $f(x) = f^n(f^{n-1}(x))$ where f^n is the n-th layer of the network and n stands for the *depth* of the model. If multiple hidden layers are present in the FFN, it is also called a deep FFN. This is also where the name *deep learning* comes from. The first layer of the network is the input, and the last layer is the output and any layer in between these two for which we do not know the true value is called a hidden layer. Moreover, the dimensionality of the hidden layers is the model's width. Finally, the FFN is called a *neural* network because it loosely mimics how the information flows through the biological nervous system, and thus each element in an FFN is called a neuron [GBC16].

FFNs can have many hidden layers, but the simplest one contains three layers: input, hidden and output.

- Input layer: takes the initial data inserted into the network. For example, if we want to input a 2-dimensional image into the network, we first reshape it into a 1-dimensional vector, and for each pixel in the image, the input layer has a separate neuron.

- Hidden layer: fully connected to each input neuron, takes in the weighted signal from the previous layer and each neuron processes it with its activation function. The activation function decides if the neuron is activated or not, after calculating the weighted sum and adding a bias [Zha+21].
- Output layer: returns the prediction of the network. The output layer is fully connected to the previous layer and has as many neurons as the task needs. For example, if the task is to determine between 3 categories from an image, the output layer's width would be 3 - one for each category. The output layer will calculate the probability that the image contains one of the three categories.

An example FFN is shown in figure 1.

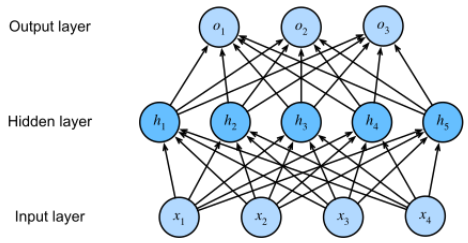


Figure 1. A FFN with 4 input neurons, 5 hidden neurons and 3 output neurons [Zha+21].

In order to make FFNs usable with input data and solve complicated regression or classification tasks, it needs to have its weights adjusted through a process called training. The FFN is shown examples of input data and their labels during the training process. Forward propagation is the process of taking x and propagating through the hidden layers and finally producing a prediction $\hat{y}$. The prediction is compared to the actual value of the input y to calculate the error of the prediction. This error is the information fed into a back-propagation algorithm to flow backwards through the network towards the input to calculate the gradient. The calculated gradients show how much each weight has to be adjusted by to minimise the error of the prediction (aka loss). The weights are adjusted after the gradients are calculated. The cycle of forward propagation and back-propagation is repeated multiple times to minimise the prediction error [GBC16; Zha+21].

With deep FFNs, more time and data are needed to train the model. However, there are multiple ways to improve the speed and performance of training. One concept is to use a pre-trained model, a model that has been trained on a large-scale general dataset (e.g. Imagenet [Den+09]), instead of training from scratch with randomly initialised weights. The idea behind pre-training is to take advantage of the weights learned from a previous task, which are closer to the optimal than randomised weights, and further train them (aka fine-tuning) to the task at hand. Fine-tuning these days is common practice in

research settings for many tasks to speed up training, use fewer data and achieve more robust models [Dev+18; Hen+20].

We also have to touch on the topic of activation functions briefly. There are many types of activation functions, but the main idea behind them is to take the weighted sums of the neurons to calculate a scalar value that will define the layer's output. In this thesis, we will cover the softmax function because the InD models include a softmax layer and softmax is used in the following sections as an intermediary step during training and its outputs can also be used for OOD detection. Softmax is most commonly used before the output layer and has the same number of neurons as the output layer. It converts the inputs into a probability distribution of the classes with the probabilities (values between 0 and 1) summing to 1 [GBC16; Bro20]. Softmax for input element x_i from an input vector x is defined as Equation 1.

$$softmax(x)_i = \frac{exp(x_i)}{\sum_{j=1}^{n}(exp(x_j)} \tag{1}$$

For ANNs with a probability distribution output (e.g. from a softmax layer) the most common loss function is cross-entropy. Cross-entropy loss measures the performance of a classification models output probability distribution to that of the actual probability distribution and our goal is to minimise it. In this thesis cross-entropy is used for training all the models on InD datasets. The most common probability distribution to use with cross-entropy is a one hot encoded vector that contains binary values, where each element corresponds to one class. The correct class is assigned the value 1, while other classes are assigned 0. We will be using categorical cross-entropy due to training models for multi-class classification ($\mathcal{L}_{CE}$) and it is defined as Equation 2, where $\hat{y}$ are the models predictions, i the i'th input and k the index of the correct class probability [KK20].

$$\mathcal{L}_{CE} = -log(\hat{y}_{i,k}) \tag{2}$$

2.2 Convolutional Neural Networks and Residual Networks

In this section, an introduction to convolutional neural networks and residual networks is given. In-depth coverage is not within the scope of this thesis.

A convolutional neural network (CNN) is a DNN originally developed for images and useful for various tasks where data can be outlaid in a gridlike pattern. The key element is a mathematical operation called convolution, hence the name of CNNs. The convolutional layer extracts a feature map from the input (e.g. image, layer) using learnable weights called filters (aka kernels) that slide across the input and extract features (e.g. lines, shapes, objects) [GBC16]. Kernels used in convolutional layers are smaller or equal to the size of the output in order to reduce the number of learnable parameters in the network [Zha+21]. Figure 2 shows an example of a convolution operation.

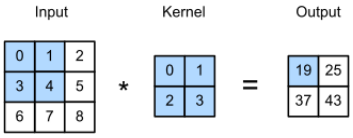


Figure 2. The 2-dimensional convolution operation. Example 2x2 output (aka feature map) of a 2x2 kernel sliding across a 3x3 input [Zha+21].

One can stack multiple layers of convolutions to get a deeper CNN. However, the vanilla deep CNNs are limited in the number of layers they could sustain because of vanishing/exploding gradients. Vanishing gradients happen when values of the loss function approach zero during back-propagation, and *vice-versa* for exploding gradients, making it impossible to train the model. Furthermore, there was the problem of degradation with deeper networks, which stemmed from the fact that by adding more layers, accuracy of the network got saturated, leading to higher training error. ResNet (aka Residual Network) architecture by He *et al.* [He+16] largely mitigates the problem of vanishing/exploding gradients by introducing the notion of the residual block. Residual blocks, in a sense, contain skip-connections (aka residual connections) of a previous layer that make inputs forward propagate faster through layers [Zha+21]. Figure 3 shows an example residual block, with a regular block of an FFN on the left and a residual block with a residual connection on the right.

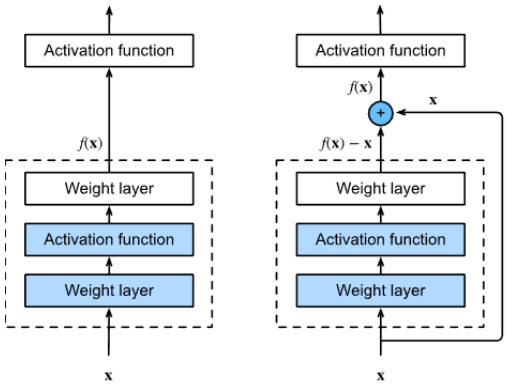


Figure 3. Regular block (left), residual block (right) [Zha+21].

CNNs are models that use convolutional operations in any of their layers, and residual networks use residual connections in their architecture. Residual connections help a deeper model train more effectively and avoid gradient exploding/vanishing and the problem of degradation. Moreover, residual blocks help propagate faster through the residual connections across layers during the forward propagation step.

2.3 Transformer

This section will give an in-depth introduction to the Transformer architecture because this is the primary building block of the Vision Transformer.

Before diving into the Transformer architecture, we have to introduce the concept of embeddings (aka tokens). An embedding is a dense low-dimensional vector representation of numbers that includes information about an object, such as a word or an image. Machine learning uses embeddings due to machine learning algorithms requiring numerical representation of data for training and deployment. Embeddings can be learned during model training as an intermediate representation of the training inputs that are most helpful to perform the task. The input data is mapped into latent space, whose main property is the representation of the input data, where semantically similar objects are near one another, making distance metrics a viable way to measure the similarity of inputs [Liu22; Gao21]. Embeddings are used in many fields of machine learning, such as machine translation, speech recognition, OOD detection and many more.

The Transformer architecture was first introduced by Vaswani *et al.* [Vas+17] for natural language processing (NLP). Recurrence based architectures have limited use in NLP due to limitations of parallelisation and memory constraints for longer sequences caused by their sequential nature. Meanwhile, the Transformer architecture negates the issues of recurrent networks by not relying at all on recurrence or convolutions to generate outputs. It relies solely on connecting the encoder and decoder through attention mechanisms. The attention mechanism allows a model to draw from the state at any preceding point in the sequence. Transformers take into account the global dependencies of input and output, which helps in better generalisation in various NLP tasks, and therefore, models based on Transformers are the models of choice in NLP [Dos+20; Tou+20].

The Transformer is based on the ideas of self-attention and Multi-Head Attention (MHA), which decide by looking at the input sequence what other parts of the sequence are important for making the prediction. Attention is a function mapping queries Q to a set of key-value pairs K and V to an output. To calculate the value of we take an input x (e.g. sequence of word vectors) and do the following calculations: $x * W_q = Q$, $x * W_k = K$ and $x * W_v = V$, where W_q, W_k and W_v are learnable weight matrices of Q, K and V. The weight matrices are learned during the training of the model through backpropagation. The idea is to use Q to find the most similar K to retrieve V. To calculate attention Q and K^T are multiplied together and divided by the square root of the dimension of the keys d_k to scale the values because softmax is sensitive to large input values and to speed up training. Softmax is used to normalise outputs and to return a probability distribution of K with the highest values belonging to the most similar Q's. Finally, the previous output is multiplied with V to retrieve the attention for the respective Q, K and V combination [Blo]. Vaswani *et al.* [Vas+17] term their attention "Scaled Dot-Product Attention" (Equation 3).

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{3}$$

Instead of performing a single attention function with d-dimensional K, V and Q, h attention function is calculated in parallel by linearly projecting Q, K and V to d_q, d_k and d_v dimensions h times. Yielding a d_v dimensional output value, and these are concatenated and projected again to result in the final values. This is done for MHA to attend to different sequence representations at different positions simultaneously. Figure 4 visualises the scaled dot product attention and MHA.

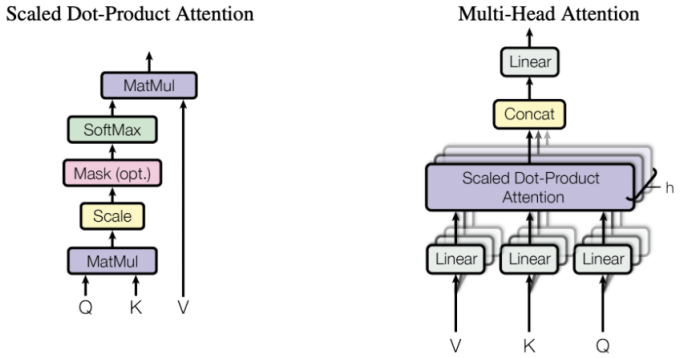


Figure 4. Scaled Dot Product Attention (left). Multi-Head Attention (MHA) consisting of h attention layers in parallel (right) [Vas+17].

Transformers are an encoder-decoder architecture where the encoder's task is to map an input sequence to a sequence of continuous representations, which are fed to a decoder. The inputs and outputs in the context of NLP are words which are transformed to embedding vectors. The decoder's task is to generate output based on the encoder's output and the decoder's output from the previous time step. During training use teacher forcing, which is to assume the prediction from the previous step is always correct to avoid recurrence. They both consist of modules that can be stacked on top of each other multiple times - most commonly 6. At the same time, the modules themselves contain MHA and Feed Forward (FF) layers. The previous step's input and output are embedded into n-dimensional space with their positional embedding. A positional embedding for the inputs is required to get the information about the relative positions of the words in the sequence since a Transformer does not use recurrence. Figure 5 shows the architecture of the encoder-decoder structure, with the encoder on the left and the decoder on the right half. The Nx is the value how many times these components are stacked on top of one another.

The encoder stack on the left side of Figure 5 is composed of multiple identical layers, with each layer consisting of two sub-layers. The first sub-layer is a MHA mechanism,

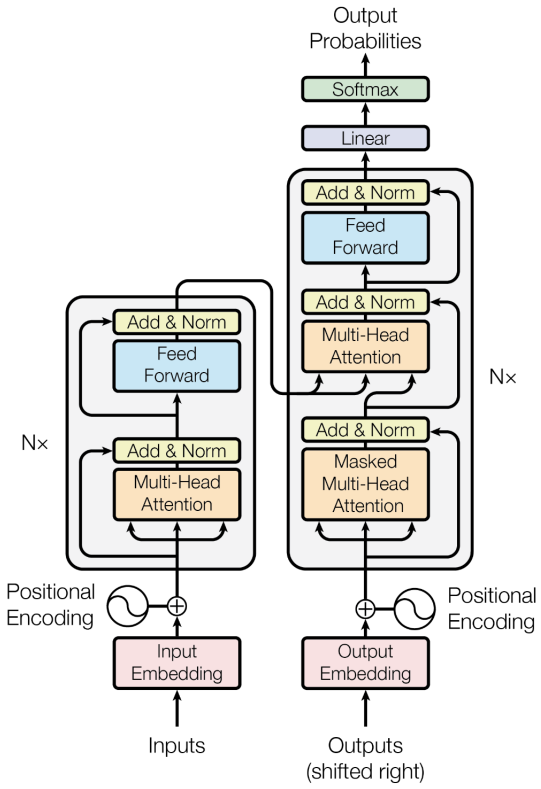


Figure 5. The Transformer - model architecture [Vas+17].

and the second is a fully connected FF network. For each layer, a residual connection goes around both sub-layers and is associated with the output of the other two layers through layer normalisation [Vas+17]. Layer normalisation (aka Layernorm) is simply normalising the output of layer activation's to zero mean and unit variance [BKH16]. Layer normalisation is done to ensure that the scale of the outputs is always the same speeding up training.

The decoder on the right side of Figure 5 shares similarities with the encoder by being composed of a stack of 6 identical layers and uses a residual connection between the sub-layers followed by LN. In addition to the two sub-layers present in the encoder stack, the decoder has a third layer, which performs MHA over the output of the encoder. The decoder stack also modifies the self-attention sub-layer by masking the subsequent positions (Masked Multi-Head Attention). This is done to ensure that predictions at position i only depend on outputs at positions less than i. Finally, to generate a prediction for the next word of the output sequence, the output of the decoder passes through a fully connected layer, followed by a softmax layer that outputs the prediction probabilities.

The Transformer architecture is the first model entirely based on attention mecha-

nisms. Compared to recurrent and convolutional based architectures, Transformer based architectures have shown significant performance increases in NLP tasks. Leading to multiple Transformer-based architectures such as BERT demonstrating breakthrough results [Dev+18]. All of this is possible due to the computational efficiency and scalability of the Transformer, making it possible to train models of unprecedented size [Dos+20].

2.4 Vision Transformer

This section gives an overview of Vision Transformer and its data-efficient variant (Data-efficient image Transformer). Data-efficient image Transformer is introduced briefly, because it will be used in the experiments.

In computer vision, Transformers, until recently, were used either in conjunction or as a part of a CNN. However, in NLP tasks, purely Transformer [Vas+17] based architectures were widely used, with tremendous success and the models of choice. The primary approach to using Transformer based architectures is to pre-train on a large dataset and then fine-tune on a smaller task-specific dataset [Dev+18]. Inspired by the Transformer's success in natural language processing, Dosovitsky *et al.* [Dos+20] experimented with applying Transformers directly on images and achieved SOTA results with a purely Transformer based architecture without using any convolutions - Vision Transformer (ViT).

The architecture of ViT [Dos+20] closely follows the original implementation of the Transformer [Vas+17], which as an input accepts 1D sequences of token embeddings. Note that the authors of ViT refer to token embeddings as patch embeddings. To make a 2D image of shape $x \in \mathbb{R}^{H \times W \times C}$, where H, W, and C denote the height, width and number of channels of the image used for the Transformer, a series of patch embeddings are constructed as follows:

- 2D image is split and reshaped into flattened 2D patches of shape $x_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$, where P denotes the resolution of each image patch and $N = HW/P^2$ the resulting number of patches, which also acts as the sequence length of the Transformer. Note that the patches are non-overlapping.

- To construct patch embeddings, flattened two dimensional patches are mapped to D dimensions with a trainable linear projection, where D denotes a constant latent vector size through all of the layers of the Transformer.

A 1D positional embedding is added to each patch embedding as an input to the Transformer encoder to retain the positional information. At the beginning of the sequence of patch embeddings, a trainable embedding is added, called the class token. The class token accumulates information from other tokens using MHA by going through the layers of the Transfomer Encoder [Tou+20]. The class tokens output from the last

layer of the Transformer encoder is the image representation, which is the input to the classification layer. The classification layer is an MLP with one hidden and linear layer at pre-training and only a single linear layer at fine-tuning [Dos+20].

The Transformer Encoder [Vas+17] takes the embedded patches as inputs and consists of alternating layers of MHA and MLP blocks. The ViT differs from the original Transformer in a sense that it only contains a Transformer encoder block and no decoder. In the Transformer encoder a Layernorm (LN) is applied before each block and residual connection. The MLP block, also referred to as FFN, is composed of two layers separated by a GeLu [HG16b] activation function. The overview of the model has been summarised in Figure 6. Note that the L_x in Figure 6 stands for the number of layers the network has and this thesis uses a ViT with 12 layers.

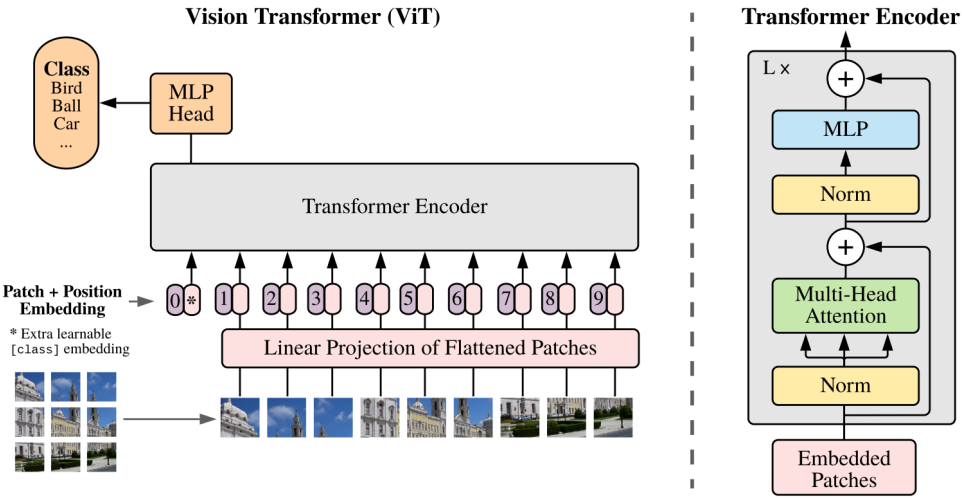


Figure 6. ViT overview with ViT architecture on the left and the Transformer Encoder section in the right. [Dos+20].

ViT does not generalise well if trained with insufficient amounts of data, requiring extensive computing time and data to be pre-trained [Dos+20]. Touvron *et al.* [Tou+20], created Data-efficient image Transformers (DeiT) to overcome this shortcoming. DeiT does not alter the architecture of ViT, but implements methods to speed up training by utilising knowledge distillation, large amounts of augmentations during training and improvements included in the `timm` library [Wig19]. The improvements come by introducing a new knowledge distillation procedure based on a distillation token, which plays the same role as a class token as a learnable token, but instead aims to reproduce the label of the class estimated by the teacher network. Figure 7 summarises the distillation procedure for DeiT.

Hinton *et al.* [HVD+15] introduced knowledge distillation after seeing the performance of ensemble models. The idea revolves around training a neural network - student - using another trainer's output - the teacher. The knowledge distillation procedure outlined by Touvron *et al.* [Tou+20] applies this methodology of training a ViT student by exploiting a teacher network - e.g. ResNet or another ViT. The authors of DeiT observed that CNNs are better teachers than other Transformer networks.

Assuming a strong image classifier as a teacher model, the distillation procedure includes a new distillation token (aka distillation embedding) concatenated to the class embedding and patch embeddings as an $N + 1$ token. Distillation embedding is used similarly to a class embedding and interacts with other embeddings through self-attention layers and is output after the last layer. The difference between class embedding and distillation embedding lies in its objective to reproduce the hard label predicted by the teacher instead of the true label (the actual class). Let Z_s, be the logits of the student model, let $\mathcal{L}_{CE}$ be cross-entropy loss on ground truth labels y and ϕ the softmax function. Let $y_t = argmax_c Z_t(c)$ be the hard decision, also known as the hard label of the teacher, where Z_t is the logits of the teacher model. The loss function used as the objective for hard-label distillation is as follows:

$$\mathcal{L}_{global}^{hardDistill} = \frac{1}{2}\mathcal{L}_{CE}(\phi(Z_s), y) + \frac{1}{2}\mathcal{L}_{CE}(\phi(Z_s), y_t) \tag{4}$$

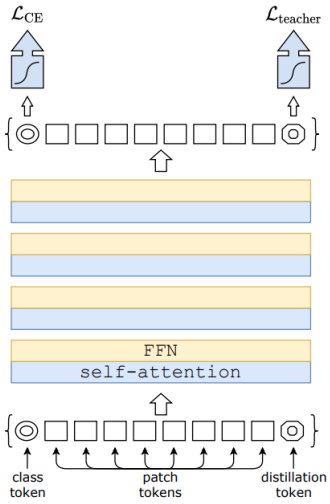


Figure 7. DeiT distillation procedure [Tou+20].

In addition, to introducing a new distillation procedure to speed up training, the authors of DeiT [Tou+20] applied extensive data-augmentations, such as Rand-Augment

[Cub+20], random erasing [Zho+20] with modifications from `timm` [Wig19] library. DeiT uses augmentations to create a data-efficient training regime. A data-efficient training regime is required due to the need of more data for training than convolution-based architectures. Finally, DeiT relies on AdamW optimizer and various data regularization methods, such as, MixUp [Zha+18], CutMix [Yun+19], stochastic depth [Hua+16] and repeated augmentation [Ber+19; Hof+20] to improve training performance.

2.5 Out-Of-Distribution Detection

This section will give an overview of the related works in OOD detection relevant to this thesis.

InD data is the data that the model was trained and tested on. It is expected that models work well with InD. On the other hand, OOD can severely hamper the algorithm's performance. There are multiple reasons why the performance is hampered, such as misclassification of inputs with high confidence that are not in any of the training classes [NYC15]. An input can be OOD due to an input introducing a new concept (e.g. new class) or input containing random noise generated by a sensor. To explain this concept more clearly, assume a cat and dog classifier, where cats and dogs are the InD and OOD is everything else that can not be classified as a cat or dog (e.g. horse, car, plane). Detecting OOD inputs is important because DNNs are more widely used in safety-critical applications, such as health-care, autonomous driving and surveillance systems, where distinguishing between InD and OOD is paramount to making safe decisions [Amo+16]. As an example, assume a situation of a self-driving car which sees a new object that it has never seen before and instead detects it as OOD and slows down or hands over the controls to the human to lower the risk of an accident. The main reason for the development of OOD detection is to make safe decisions to reduce the risk of deploying DNNs to make them aware of what they should not know.

Hendrycks and Gimpel [HG16a] came up with a simple baseline method for detecting misclassified and OOD examples with DNNs. The method uses the simple observation that correctly classified examples have a greater maximum softmax probability (MSP) than wrongly classified and OOD examples, making their detection possible. Using MSP as the confidence score is enough to detect OOD inputs. MSP is considered a baseline method and is simple to implement [FRL21].

Another popular method is to use a pre-logit layer (aka embedding, feature vector, latent space embedding) and a distance metric that will compare the samples embedding to class conditional embeddings of any softmax classifier. One needs an InD dataset with a trained softmax classifier to get class conditional embeddings. Then extract the model embedding for every InD sample and take the average embedding for each class. Using this method, one can apply distance metrics in latent space to calculate the distance from the inputs embedding to the nearest class embedding. The main idea is that InD embeddings should be closer to one another than OOD embeddings. Lee *et al.* [Lee+18]

showed that using Mahalanobis distance for OOD detection in embedding space can yield good results. Mahalanobis distance is *de facto* standard in OOD detection. To use Mahalanobis distance, one fits a Gaussian distribution to the embeddings and calculates a shared covariance matrix.

Recent improvements in OOD detection stemmed from two methods called outlier exposure and contrastive learning. Outlier exposure [HMD19] is the idea of using a large dataset of known outliers and training a model to predict a uniform distribution over the label of these inputs. It is implemented by adding uniform distribution output to the learning objective, such as cross-entropy to known outliers. Outlier exposure can improve model calibration and OOD detection performance. Furthermore, it is also easy to implement due to the availability of labelled data and being a computationally inexpensive method to apply. Meanwhile, if one requires a method that does not have access to training examples explicitly labelled OOD, one can use contrastive learning [Win+20]. The main idea of contrastive learning is to train a model in which similar objects have similar embeddings while repelling dissimilar objects. In other words, InD objects should be similar to one another and nearby one another in embedding space while repelling OOD data.

Winkens *et al.* [Win+20] introduced not only contrastive learning, but also introduced the distinction between near-OOD and far-OOD. They considered near-OOD tasks to be more challenging than far-OOD tasks. For example, a model trained on CIFAR-10 (10 classes: horse, ship, truck, aeroplane, automobile, bird, cat, deer, dog, frog) would consider detecting street signs (SVHN) a far-OOD task. Meanwhile, near-OOD to CIFAR-10 would be CIFAR-100 because it contains semantically similar classes, such as mammals and vehicles. The distinction between near-OOD and far-OOD is also evident in the SOTA performances of OOD detection models. CIFAR-10 to SVHN and CIFAR-100 to SVHN can achieve AUROC close to 99%, while CIFAR-100 to CIFAR-10 until the advent of ViT in OOD detection was SOTA AUROC of 86% [FRL21].

Hendrycks *et al.* [Hen+20] was one of the first to show the robustness of pre-trained Transformers by utilising BERT to improve OOD detection in NLP. They showed that Transformers are more effective at OOD detection than previous models, which were frequently worse than chance. In addition, they showed that Transformers are robust, larger Transformers do not always improve results, yet diverse pre-training data can improve OOD detection and robustness.

While Transformer based architectures became the *de facto* standard in various NLP tasks, the use of Transformers was not widespread in vision benchmarks. Until the advent of ViT, the most widespread type of models in vision benchmarks were CNNs, such as ResNet. In addition, the development centred around the inductive biases of CNNs, such as local context. However, after the release of ViT [Dos+20] Koner *et al.* [Kon+21] discovered that ViT could reach SOTA OOD detection results out-of-the-box without any outlier exposure or contrastive learning. They argued the performance gains were

from the global contextualisation of Transformers and showed through experiments the superiority of ViT compared to ResNets with more parameters. Fort *et al.* published a similar paper using ViT by using outlier exposure and showed that the performance could be improved even more. With only fine-tuning a ViT to CIFAR-100, the performance of OOD detection of CIFAR-100 to CIFAR-10 is 96%, but using ten images per outlier class can reach 99% AUROC. The performance gains in OOD detection using VIT are huge, considering that the SOTA performance of a similar setup using CNNs reached 86% AUROC.

To better understand why large-scale pre-training and fine-tuned Transformers have improved performance, we must look at the embeddings of the model. The below Figure 8 [FRL21] is a two dimensional PCA projection of the embedding space for three models. The leftmost figure is a ResNet that has not been pre-trained but trained on CIFAR-100, the centre figure is a ViT pre-trained on ImageNet but not fine-tuned, and the rightmost figure is a ViT pre-trained on ImageNet and fine-tuned on CIFAR-100. The figure's colour indicates the Mahalanobis OOD score, with dark blue indicating a low OOD score and white high OOD score. The figure uses two InD classes from CIFAR-100 *sunflowers* (black plus), *turtle* (yellow plus) and OOD class from CIFAR-10 *automobile*. The figure shows that Resnet has difficulty distinguishing between the different classes but shows that the angle in which class members are from each other points that they have different directions in 2-dimensional PCA. Furthermore, ViT without fine-tuning seems to have already three distinguishable classes but a low OOD score. The rightmost figure shows that ViT fine-tuned with distinguishable classes assigns high Mahalanobis OOD scores to OOD inputs.

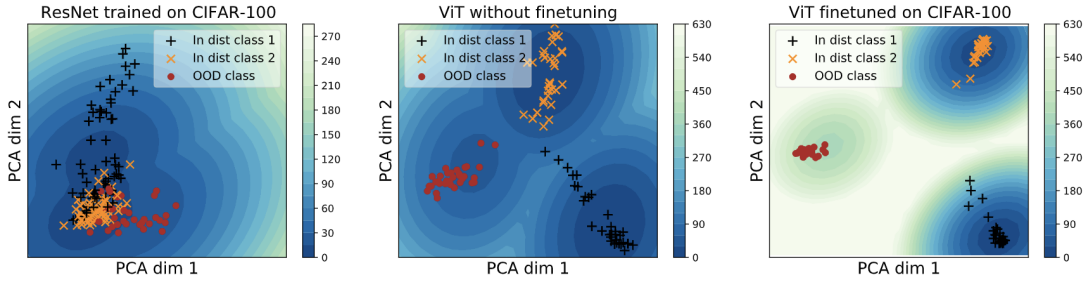


Figure 8. A 2-dimensional PCA projection of embedding space of three models (trained Resnet, pre-trained ViT, pre-trained and fine-tuned ViT), with 2 InD classes and 1 OOD class. Color denotes the Mahalanobis score [FRL21].

In conclusion, the primary method for OOD detection to achieve SOTA results is to use the embedding layer of a ViT to calculate the Mahalanobis OOD score using the class conditional embeddings of the model and a shared covariance matrix. The main improvements in OOD detection come from getting a more discriminative embedding

wherein InD objects are nearer one another while repelling OOD objects. Fort *et al.* [FRL21] and Koner *et al.* [Kon+21] showed that ViT has, after fine-tuning to a specific InD dataset, a discriminative embedding out-of-the-box without the need for outlier exposure or contrastive learning. Although, Fort *et al.* indicated that outlier exposure would have a positive impact on OOD detection.

3 Methodology

The methodology chapter introduces the essential methodological concepts used in this thesis. It is largely inspired by Lee *et al.* [Lee+18] and Koner *et al.* [Kon+21]. The main concept is that semantically similar objects such as images from the same class (e.g. cats) are very close in the latent embedding space of a softmax classifier. Meaning similar object embeddings should cluster together, and if one takes the mean of the object embeddings, the distance of the same class objects to the mean should be less than with objects of other classes (e.g. cats *vs.* cars) or random noise (e.g. Gaussian noise).

This concept of similar objects in latent embedding space being close to one another can be used with any trained softmax classifier as an embedding extractor (aka feature extractor). Data used for training and data that the model should be able to classify correctly with high confidence is considered InD data (e.g. cats and dogs). However, data that the model has not been trained to classify (e.g. cars and trucks) is considered OOD. Distance metrics can be used to measure the distance of an input embedding (can be either InD or OOD) to the InD class mean embeddings. The most popular distance metric is Mahalanobis distance because it changes the idea of comparing a point-to-point distance to a point-to-Gaussian distribution distance. However, the whole idea is built upon the quality of the embeddings of the softmax classifier, and Koner showed that pre-trained and fine-tuned ViT has much better performance in OOD detection than similarly trained CNNs from scratch. Moreover, initial evidence provided by Koner *et al.* showed that ViT could be robust to any distance metric [Kon+21; Lee+18].

Based on these ideas, we delve into using a pre-trained ViT and fine-tune it to an InD dataset and then use it as an embedding extractor. Using the InD labels of different classes, we can calculate the class mean embeddings and other statistics, such as class variances and covariance. Then we can use any sample (can be InD or OOD) and calculate the distance using any distance metric to the class mean embedding. Moreover, using the same idea that similar objects are near one another means that InD objects should have less distance to class mean embeddings than OOD samples. This allows us to use thresholds to classify objects in latent embedding space, either InD or OOD. In order to measure the performance of how well OOD detection is performed, we use AUROC performance and multiple InD and OOD datasets. Furthermore, we introduce a new experimentation method called mixer-images to see how well different models and distance measures behave with InD to OOD distribution shift in a controlled manner.

3.1 Datasets

We designate datasets either as in-distribution (InD) or out-of-distribution (OOD). InD datasets are used for training models and include CIFAR-10 [KH+09], CIFAR-100 [KH+09] and Imagenet-30 (IM-30) [Hen+19] datasets. CIFAR-10 is one of the most widely used machine learning training datasets, which consists of 60K 32x32 RGB

images, with 10 classes of 6000 images per class. CIFAR-100 is similar to CIFAR-10 but has 100 classes with 600 images each per class. Both datasets are split into 50K training and 10K testing images. It has to be noted that CIFAR-10 and CIFAR-100 do not have any overlapping classes, but some have similar attributes or concepts (e.g. CIFAR-10: 'truck', CIFAR-100: 'pickup-truck'). Thus, the semantical closeness of CIFAR-10 and CIFAR-100 poses the most difficult challenge of near-OOD if one uses one for InD and the other dataset as OOD. Another InD dataset is IM-30 which is a subset of ImageNet [Den+09]. We use IM-30 to compare OOD detection performance in a situation where models have near perfect classification accuracy, because the models have been pre-trained on ImageNet and should have very high classification accuracy with IM-30. The dataset consists of 42K RGB with 30 classes and 1400 images per class. Each class has 1300 training images and 100 test images. Although ImageNet has 50 test images per class, the authors of the IM-30 dataset have expanded and collected an additional 50 images for each class for an expanded test set.

There are two sets of OOD datasets, one used as the OOD dataset for CIFAR-10/-100 and the other set for IM-30 trained models. OOD datasets only contain their respective test sets because OOD datasets will be not used for training models. OOD datasets used for CIFAR-10/-100 are as follows:

- Street View Housing Number (SVHN) [Net+11]: 26K 32x32 RGB images of 10 classes. It consists of ten numbers (from 0 to 9) cropped from images of house number plates.
- Large-scale Scene UNdesrstanding resized (LSUN_r) [YA19]: 10K RGB images of 10 classes. It consists of 10 scene categories, such as an outdoor church, bedroom, dining room and more, with 1000 images per category. It consists of a subset of LSUN [Yu+15] images resized to 32x32. 32x32.
- Imagenet-Resize (Imagenet_r) [CLH17]: 10K 32x32 RGB images of 200 classes with 50 images per category. It consists of a subset of Imagenet images resized to 32x32.

And the following OOD datasets are used if the model is trained with IM-30 as InD:

- Places-365 [Zho+17]: 18K 256x256 RGB images of 365 classes. It comprises 365 scene categories and environments encountered globally, with 50 images per category.
- Describable Texture Dataset (DTD) [Cim+14]: 1880 300x300 to 640x640 RGB images of 47 classes. DTD is a texture database based annotated using crowd sourcing.
- Stanford Dogs (Dogs) [Kho+11]: 22K various sizes of RGB images various sizes of 120 classes. It consists of annotated images of dogs belonging to 120 breeds.

- Food-101 [BGV14] : 25K 512x512 RGB images of 101 classes. Pictures of 101 types of food which have been data-mined from the internet.
- Caltech-256 [GHP07]: 30K images of various sizes from 257 classes. Pictures from various object categories ranging from grasshopper to tuning fork.
- Caltech-UCSD Birds-200-2011 (CUB) [Wah+11]: 12K images of various sizes from 200 classes. It consists of images of 200 different types of bird species annotated using crowd sourcing.

Table 1 summarises the main properties of each dataset.

Type	Name	Number of Images	Classes	Images per class
InD	CIFAR-10	60000	10	6000
	CIFAR-100	60000	100	600
	IM-30	42000	30	1400
OOD	SVHN	26000	10	2600
	LSUN_r	10000	10	1000
	Imagenet_r	10000	200	50
	Places-365	18250	365	50
	DTD	1880	47	40
	Dogs	20580	120	180
	Food-101	101000	101	1000
	Caltech-256	30607	256	>80
	CUB	6038	200	∼30

Table 1. Summary of the datasets.

3.2 Distance measures

In this subsection, we will introduce the distance metrics (sometimes referred to as similarity measures) used in this thesis to calculate pairwise distances in latent space to InD or OOD samples. A short overview and relevant notation are given for every distance metric used in this thesis.

The distance function between 2 numerical vectors $x = (x_1, , ..., x_n) \in \mathbb{R}$ and $y = (y_1, ..., y_n) \in R$ is the function $d(x, y)$ that defines the distance between both vectors [ED20]. In this thesis we will consider the use of the following distance measures or similarity measures: Minkowski, Standardised Euclidean, Mahalanobis distance and Cosine similarity.

Minkowski distance can be considered as a family of distance metrics and includes three distance metrics that are special cases: Euclidean, Chebyshev and Manhattan distance. It is defined as Equation 5, where p is a positive value. The special cases for

Minkowski are $p = 1$ for Manhattan distance, $p = 2$ Euclidean distance and $p = \infty$ for Chebyshev distance [ED20].

$$d_{minkowski} = \sqrt[p]{\sum_{i=1} |x_i - y_i|^p} \tag{5}$$

Standardised Euclidean distance is a variant of Euclidean distance where each variable is weighted with a separate variance and is used when the direction of the vector is meaningful but the magnitude is not. Standardised Euclidean is defined as Equation 6, where $V = \frac{1}{n}\sum_{i=1}(x^i - \mu)^2$ is the variance vector, and $\mu = \frac{1}{n}\sum_{i=1} x_i$ is the average of all x. V is computed over all the i'th components of the points [Cho+15].

$$d_{s\_euclidean} = \sqrt{(x-y)V^{-1}(x-y)^T} \tag{6}$$

Mahalanobis distance is a metric to calculate the distance between two points in multivariate space. Mahalanobis distance is equal to Euclidean distance if the input vectors are not correlated but differ when the input vectors correlate. Mahalanobis distance is defined as Equation 7, where $C = \frac{1}{n}\sum_{i=1}(x_i - \mu)(x_i - \mu)^T$ is the covariance matrix of the inputs. Mahalanobis distance is widely used in OOD detection as a distance metric [Lee+18; Kon+21; FRL21] and is the current SOTA. It is similar to Standardised Euclidean distance in a sense that the diagonal of the covariance matrix is V.

$$d_{mahalanobis} = \sqrt{(x-y)C^{-1}(x-y)^T} \tag{7}$$

Cosine similarity, also known as the angular distance between vectors, is the cosine of the angle between two n-dimensional vectors in an n-dimensional space. It is a widely used distance metric for computing similarity as part of a recommendation query or in NLP to determine how similar documents are to each other irrespective of their size. Cosine similarity is defined as Equation 8 [ED20].

$$d_{cosine} = 1 - \frac{\sum_{i=1} x_i y_i}{\sqrt{\sum_{i=1} x_i^2}\sqrt{\sum_{i=1} y_i^2}} \tag{8}$$

3.3 Performance Metrics

This section will overview the performance metrics used in this thesis - accuracy and area under the receiver operating characteristic (AUROC). To measure the performance of the models on InD classification, we use accuracy. Meanwhile, AUROC is used to evaluate the performance of OOD detection.

OOD detection can be thought of as a binary classification problem, where the actual class is positive if the sample is InD or negative, if the sample is OOD. Binary classification can lead to four possible types of predictions [SR15]:

- True positive (TP): correct positive prediction;
- False positive (FP): incorrect positive prediction;
- True negative (TN): correct negative prediction;
- False negative (FN): incorrect negative prediction.

The four possible outcomes of the predictions of a binary classifier can be used to calculate Accuracy: $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$, True positive rate (TPR) (Also known as sensitivity or recall): $TPR = \frac{TP}{TP+FN}$ and False Positive rate (FPR) (also known as false alarm rate): $FPR = \frac{FP}{TN+FP}$. TPR and FPR are used to calculate AUROC.

AUROC is calculated as the area under the Receiver Operating Characteristics (ROC) curve, which shows the trade-off between TPR and FPR across different decision thresholds. ROC curve shows pairs of TPR and FPR values calculated at all possible threshold scores. A random classifier, which is also defined as the baseline, would be a diagonal line from the origin to the top right corner of the plot. A perfect classifier would form two straight lines - from the origin to the top left corner and further to the top right corner. It is important to note that a meaningful ROC curve would lie in between a random classifier and a perfect classifier. Using the ROC curve, one can calculate AUROC, which is simply the area under the ROC curve. The AUROC can have values between 0 and 1, but actual scores lie between 0.5 and 1. An AUROC of 0.5 will be equal to a random classifier, and an AUROC of 1 is a perfect classifier. The notion of AUROC and ROC is summarised in Figure 9.

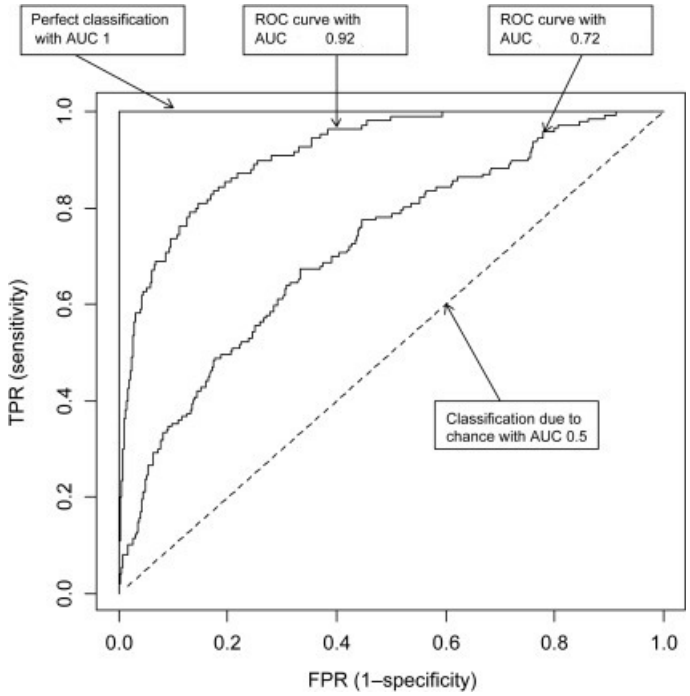


Figure 9. Example AUROC curve of 4 binary classification models [PT17].

There are multiple reasons to use AUROC instead of accuracy for OOD detection:

- A single threshold may not scale across all the data sets, and AUROC considers all thresholds while accuracy uses one threshold.
- AUROC is not sensitive to class imbalance.

In addition, AUROC is the most popular performance metric used in all OOD detection papers. Although, some papers also use the TPR of the 95th percentile (TPR95) and FPR of the 95th percentile (FPR95). This thesis will concentrate on using AUROC.

3.4 Out-Of-Distribution Detection

As mentioned in Section 3.1 the datasets are divided into two sets - InD and OOD. We trained image classification models using InD samples. These trained image classification models can be used as feature extractors given an input. The input to the feature extractor can be from any distribution, both InD and OOD. However, we expect InD samples to stay in the InD distribution and OOD samples, where one or more attributes are different, lie in a distribution that has a sizeable distributional shift from InD samples. To quantify this shift, we first use a trained feature extractor to retrieve the model embedding (aka feature vector, pre-logits layer). Then we compute the embedding similarity between

the sample and the nearest InD class mean using a distance metric (see Section 3.2). In an ideal scenario, the similarity for OOD should be much less than InD. Furthermore, the model's softmax confidence for OOD should be considerably lower than an InD sample allowing simple thresholds to distinguish between InD and OOD samples. The methodology is largely inspired by [FRL21] and [Kon+21]. Algorithm 1 describes the process of retrieving the class mean, class Variance and shared covariance matrix that are required to calculate the distance metrics in embeddings. Shared covariance instead of class covariance is calculated due to numerical instability and underfitting caused by too few samples for each class in some datasets. For example, CIFAR-100 has 500 training samples per class, but the ViT extracts an embedding vector of size 768 for each input. To calculate the shared covariance, we subtract the inputs of the same class, and its class mean and calculate covariance once. This means that for Mahalanobis distance, we use independent class means but share a covariance with all classes [Ren+21]. Algorithm 2 defines the process of OOD detection using various distance metrics with embeddings and MSP. Finally, the notation is described below:

- Assume we have an InD data sample $x_{in} \in X_{in}$ and its class label $y_{in} \in Y_{in} = \{1, ..., C\}$ in dataset $D_{in} = (X_{in}, Y_{in})$;
- A model is fine-tuned with a training set D_{in}^{train} without any access to OOD.
- Learned feature X_{feat} is obtained using a feature extractor $F_{feature}$ by $X_{feat} = F_{feature}(X_{in})$;
- Feature classifier $F_{classifier}$ is used to obtain posterior probabilities $P(y = c|x_{feat})$, where $x_{feat} \in X_{feat}$ and $y \in Y = F_{classifier}(X_{feat})$;
- OOD data sample $x_{out} \in X_{out} \notin X_{in}$, with its class labels $y_{out} \in Y_{out} = \{C + 1, ..., C + O\} \not\subset Y_{in}$ in dataset $D_{out} = (X_{out}, Y_{out})$;
- To quantify if a sample is OOD, calculate a distance using distance metric d_{metric} between a sample x_{in} or x_{out} and the mean of each classes X_{feat}^{in}.
- Sample is classified OOD if distance is more than threshold $t_{distance}$ to its nearest class.
- The performance is evaluated using a test set which contains D_{in}^{test} and D_{out}^{test} with metric AUROC.

Algorithm 1: Precomputing values using InD data

Data: training samples x_{in}^i and training labels y_{in}^i for $i = 1 : n$
Result: Class means μ_c, class variance V_c and shared covariance matrix $\mathbf{\Sigma}$

$x_{feat}^i = F_{feature}(x_{in}^i)_{i=1:n}$
for *each class* $c \leftarrow 1$ *to* C **do**
  $\{x_{feat}^{c,i}\}_{i=1:n_c} = \{x_{in}^{c,i} | y_{in}^{c,i} = c\}_{i=1:n_c}$, where $n_c = |\{x_{in}^{c,i} | y_{in}^{c,i} = c\}|$
  compute mean: $\mu_c = \frac{1}{n_c} \sum_{i=1}^{n_c} (x_{feat}^{c,i})$
  compute variance: $V_c = \frac{1}{n_c} \sum_{i=1}^{n_c} (x_{feat}^{c,i} - \mu_c)^2$
compute shared covariance $\mathbf{\Sigma} = \frac{1}{n} \sum_{c=1}^{C} \sum_{i=1}^{n_c} (x_{feat}^{c,i} - \mu_c)(x_{feat}^{c,i} - \mu_c)^T$

Algorithm 2: OOD detection using a distance metric [Kon+21] or MSP

Data: InD shared covariance $\mathbf{\Sigma}$, InD variance V, InD class mean embeddings μ_c, test sample x_{test}
Result: Is x_{test} an outlier ?

$x_{feat}^{test} = F_{feature}(x_{test})$
if d_{dist} *is mahalanobis* **then**
  $distance = \min_c(d_{dist}(x_{feat}^{test}, \mu_c, \mathbf{\Sigma}))$
if d_{dist} *is s_euclidean* **then**
  $distance = \min_c(d_{dist}(x_{feat}^{test}, \mu_c, V_c))$
else
  $distance = \min_c(d_{dist}(x_{feat}^{test}, \mu_c))$
$conf = \max_c(softmax(F_{classifier}(x_{feat}^{test})$
if $distance > t_{distance}$ *OR* $(conf < t_{conf})$ **then**
  x_{test} is an outlier
else
  x_{test} is not an outlier

3.5 Mixer Images

Mixer images are images where an OOD image is mixed into an InD image. As in OOD detection, OOD samples should lie in a distribution that has a sizeable distributional shift from InD samples. To visualise this shift and how models react with distributions changes diverging from InD, OOD information is merged into the InD image with increasing ratio until it becomes an OOD image. A random image is taken from InD test set and a random image is taken from the OOD set. The images are mixed with either MixUp [Zha+18], CutMix [Yun+19] or RandomMix with a ratio m, which we call Mixer Ratio. If $m = 0$, then no OOD image information is in the mixer image, and it is equal to the

InD image. If $m = 0.5$, then 50% of the InD image information has been replaced with the information of the OOD image. If $m = 1$, then no InD information is left in the image, and it is entirely OOD. MixUp blends the InD and OOD images with the Mixer ratio, while CutMix cuts and pastes an OOD image onto the InD image with a size equal to the Mixer ratio. RandomMix randomly takes pixels from the OOD image at the same location as the InD image and cuts and pastes them onto the InD image with a size equal to the Mixer ratio. An example of the different mixing logic can be seen in Figure 10. In the example, an InD image of class *frog* from CIFAR 10 is mixed with an OOD image *ray* from CIFAR-100 with increasing mixer ratios.

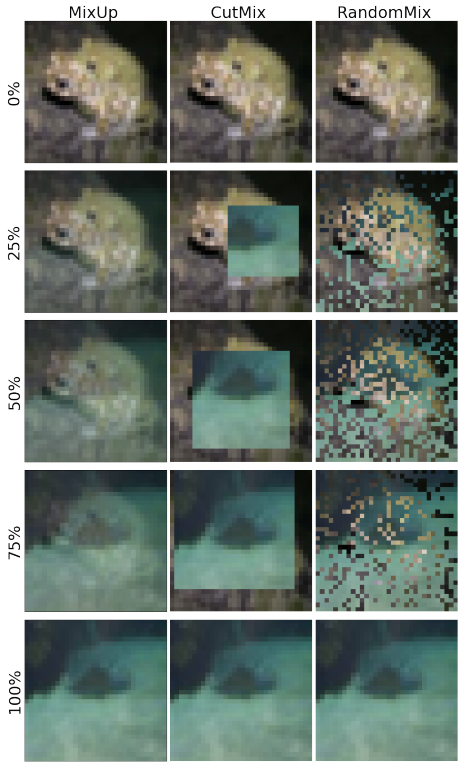


Figure 10. Mixer images example. InD CIFAR-10 frog mixed with an OOD CIFAR-100 ray at various mixer ratios. Left is MixUp, the centre is CutMix and right is RandomMix mixer logic.

To calculate the distance between the InD image and the mixer image, the below steps are done:

- Assume we have an InD data sample $x_{in} \in X_{in}^{test}$ and a OOD data sample $x_{out} \in X_{out} \notin X_{in}$.
- With x_{in} and x_{out} with ratio m to create a mixed image x_m, where m is equal to

how much info of x_{in} is replaced by x_{out}.

- Using feature extractor $F_{feature}$ return $x_{feat}^{in} = F_{feature}(x_{in})$ and $x_{feat}^{m} = F_{feature}(x_m)$.
- Calculate the pairwise distance between x_{feat}^{in} and x_{feat}^{m}.

We use mixer images to create new OOD datasets that are a mix of InD and OOD test datasets and evaluate the performance of OOD detection using the methodology described in Section 3.4. This determines how sensitive a distance metric and embedding are to distribution shift at scale. It is assumed that there should be a gradual rise in OOD detection performance with the increase of m. We also use different mixing logic and calculate the results for each one separately. In order to mix datasets into one another and calculate the performance of OOD, the following methodology is used:

- Assume we have an InD data sample $x_{in} \in X_{in}^{test} \in D_{in}^{test}$ that comprises of InD data samples and OOD data sample $x_{out} \in X_{out}^{test} \in D_{out}^{test}$
- For each x_{in} mix into them a random x_{out} data sample taken from a uniform distribution with mixer ratio m to create a new dataset D_{m}^{test}
- Calculate OOD performance using Algorithm 2.

3.6 Setup

DeiT Base-16 [Tou+20] with embedding size 768 and patch size 16 is used for all experiments regarding ViT. ResNet50 [Hua+16] with a feature vector size of 2048 is used for all experiments as the comparison CNN. All of the models have been pre-trained on Imagenet. For fine-tuning to InD distribution datasets (see Section 3.1), all models are trained for 50 epochs with supervised cross-entropy loss along with stochastic gradient descent [Rud16] as an optimiser, a batch size of 256 and image size of 224x224. We use a learning rate of 0.01 with a cyclic learning rate scheduler [Smi17] and a weight decay of 0.0005. We use a random resize crop, random horizontal flip, and normalisation for data augmentations during fine-tuning. For testing, the input is only resized and normalised.

4 Experiments

The thesis's main objective is to determine if there is any distance metric that works better for OOD detection with ViT's embeddings. Furthermore, seeing which distance metrics work better for OOD detection using ViTs will give us insights into the embedding space's properties. We analyse and explain how different distance metrics react to distribution shift using ViT's embeddings. We compare the results of ViT to a ResNet50 model to explain and visualise why ViTs might be better at OOD. This chapter describes OOD detection experiments and analysis of mixer images in detail.

4.1 Baseline Out-Of-Distribution Detection

The baseline method uses a softmax classifier pre-logit embeddings to calculate distances (aka OOD score) between InD class mean embeddings and the input. For the inputs, we use the images from InD or OOD datasets (see Section 3.1). We define OOD detection as a binary classification problem because we have two distinct groups of inputs (InD and OOD) and can use a threshold to classify InD and OOD. For the baseline models we use the OODformer framework [Kon+21] with an Imagenet pre-trained DeiT Base-16 architecture [Tou+20] and a similarly pre-trained ResNet50 [He+16] fine-tuned to an InD dataset. Table 2 summarises the test set accuracy scores of their respective InD datasets. The distance metric used in the baseline method to calculate distances in the pre-logit embedding space of a model is Mahalanobis distance (see Section 3.2) using a shared covariance matrix (see Section 3.4). We use Mahalanobis distance because it is the *de-facto* standard used for OOD detection in the last few years [Kon+21; FRL21; Lee+18].

Dataset/Model	ResNet50	DeiT B-16
CIFAR-10	96.5	98.3
CIFAR-100	83.1	88.7
IM-30	98.1	99.3

Table 2. Test set accuracy of models.

To conduct this experiment, three DeiT Base-16 models were fine-tuned with InD datasets CIFAR-10, CIFAR-100 [KH+09] and IM-30 [Hen+19] using the setup described in Section 3.6. OOD dataset for InD datasets CIFAR-10 and CIFAR-100 are resized Imagenet [CLH17], LSUN [Yu+15], SVHN [Net+11]. Furthermore, OOD dataset is CIFAR-100 if InD is CIFAR-10 and *vice versa*. For the model fine-tuned with IM-30 [Hen+19] the OOD datasets are: Places-365 [Zho+17], DTD [Cim+14], Dogs [Kho+11], Food-101 [BGV14], Caltech-256 [GHP07] and CUB-200 [Wah+11]. For OOD datasets we use the validation or test set depending on which is available. More

detailed description of the datasets is in Section 3.1. Same was repeated for ResNet50 models to get a comparison score of the same methods applied to CNNs.

To measure the performance of the predicted OOD scores, we use AUROC (see Section 3.3) because OOD detection is a binary classification task with classes of InD and OOD. For the use of AUROC, the InD label is attached to InD data and the OOD label to OOD data in the experiment. Table 3 summarises the baseline OOD detection performance of ViT and ResNet.

InD	**OOD**	**ViT**	**ResNet50**
CIFAR-10	Imagenet_r	**98.5**	87.1
	LSUN_r	**99.5**	86.2
	SVHN	**97.5**	94.9
	CIFAR-100	**97.9**	79.0
CIFAR-100	Imagenet_r	91.2	**94.0**
	LSUN_r	92.1	**94.2**
	SVHN	93.4	**97.2**
	CIFAR-10	**91.2**	71.3
IM-30	CUB	**100.0**	79.8
	Caltech-256	**96.4**	76.6
	Dogs	**100.0**	89.5
	DTD	**99.3**	97.7
	Food-101	**98.4**	81.0
	Places-365	**98.6**	75.5

Table 3. Baseline OOD detection AUROC of OODformer for ViT and ResNet50 using Mahalanobis distance. In bold are the best result for that InD and OOD combination.

Table 3 shows that ViT outperforms ResNet50 on OOD detection in most cases and only succumbs when InD is CIFAR-100 and OOD are LSUN, SVHN and Imagenet_r. When CIFAR-10 is OOD for InD CIFAR-100 and *vice versa*, there is a substantial gap between ViT and ResNet50. Near-OOD detection between CIFAR-10 and CIFAR-100 is considered to be the most challenging task since they share many common attributes in similar classes. The worse results of near-OOD detection for Resnet50 indicate that similar classes are very near each other with similar distributions, leading to overlapping class embeddings for CIFAR-10 to CIFAR-100. This result coincides with the findings of Fort *et al.* [FRL21], who showed that ResNet embeddings, when projected using PCA, lead to overlapping clusters of class embeddings. Whereas for far-OOD, where the InD and OOD datasets do not share many common attributes, ResNet50 achieves a marginal gain (InD CIFAR-100 to OOD Imagenet_r, LSUN_r, SVHN) compared to ViT.

For all other InD and OOD combinations, ViT had a substantial edge on OOD detection and was more stable than ResNet50. It is stable in that it achieved similar

results regardless of InD and OOD combination, indicating a better generalisation power of ViT. Additionally, results indicate that ViT achieves a discriminatory embedding that improves OOD detection. ViT had 90% and more AUROC in all cases, whereas, ResNet50 struggled with near-OOD and when InD was IM-30 - except when DTD was OOD.

We get better results using ResNet when compared to Koner *et al.* [Kon+21], because we use a pre-trained ResNet and resize all images to 224. In their work, they trained a Resnet from scratch and for InD CIFAR-10 and CIFAR-100 resized images to 32x32. This indicates that upsampling and pre-training have a positive impact on OOD detection.

In conclusion, ViT has better accuracy in classification in all InD datasets. The classification performance also matches the OOD detection performance by showing that ViT outperforms ResNet in most cases using the model's latent space embedding and Mahalanobis distance for calculating distances between sample and InD class means for OOD detection. Furthermore, the baseline ViT has better near-OOD detection by a larger margin. This is confirmed by having noticeably higher AUROC score in OOD detection for both CIFAR-10 to CIFAR-100 and *vice versa* coinciding with the claims made by Koner *et al.* [Kon+21] and Fort *et al.* [FRL21]. Moreover, it confirms the claim of Transformer being the better feature extractor out-of-the-box.

4.2 Out-Of-Distribution Detection with Selected Distance Metrics

This experiment examines the effects of OOD detection performance using different distance metrics. The experiment setup and models used are the same as the baseline but differ in using other distance metrics in the embedding space of the models for OOD detection. We use Chebyshev, Euclidean, Standardised Euclidean, Cosine and Mahalanobis to compare embedding space. For comparison with distance metrics using the embedding space, we also include the results of Maximum Softmax Probability (MSP). For all of the considered distance metrics, see Appendix I. Furthermore, in Appendix I, the experiments have been repeated on ResNet50.

We choose Chebyshev to see if ViT OOD data moves away from InD in one primary dimension in embedding space. Euclidean and Cosine distances are commonly used in OOD detection. Cosine distance can give us insight into if the OOD embeddings, when considered as vectors, are pointing in the same direction as InD. Furthermore, Koner *et al.* [Kon+21] state that Cosine distance should have slightly lower values than Mahalanobis, so we want to confirm this claim. Additionally, we chose Standardised Euclidean distance to see if removing class variance impacts OOD detection positively. Finally, we consider softmax MSP as a method that only uses the confidence score of the model for OOD detection. Table 4 summarises the results with the average rank for each InD and OOD combination at the bottom. Figure 11 is the visualization of Table 4.

InD	OOD	MSP	Chebyshev	Cosine	Euclidean	Standardized Euclidean	Mahalanobis
CIFAR-10	Imagenet_r	94.9	94.3	96.1	96.8	97.6	**98.5**
	LSUN_r	96.5	96.5	98.1	98.6	98.5	**99.5**
	SVHN	99.4	94.0	**99.9**	97.4	93.4	97.5
	CIFAR-100	96.5	92.3	97.7	97.7	96.8	**97.9**
CIFAR-100	Imagenet_r	87.5	87.5	92.1	**92.8**	89.4	91.2
	LSUN_r	87.1	90.5	91.4	**92.9**	89.1	92.1
	SVHN	85.9	88.8	92.0	91.5	87.5	**93.4**
	CIFAR-10	83.8	80.4	89.5	90.1	91.1	**91.2**
IM-30	CUB	99.6	96.6	99.7	99.7	98.2	**100.0**
	Caltech-256	**96.7**	93.4	**96.7**	**96.7**	95.7	96.4
	Dogs	99.6	96.6	99.8	99.8	98.4	**100.0**
	DTD	98.5	96.8	99.0	99.1	98.4	**99.3**
	Food-101	97.7	95.5	97.7	98.2	98.0	**98.4**
	Places-365	98.7	95.9	98.8	**98.9**	98.2	98.6
Average Rank		4.4	5.6	2.8	2.3	4.2	**1.8**

Table 4. ViT OOD detection AUROC results of selected distance metrics. In bold are the best results for that InD and OOD combination.

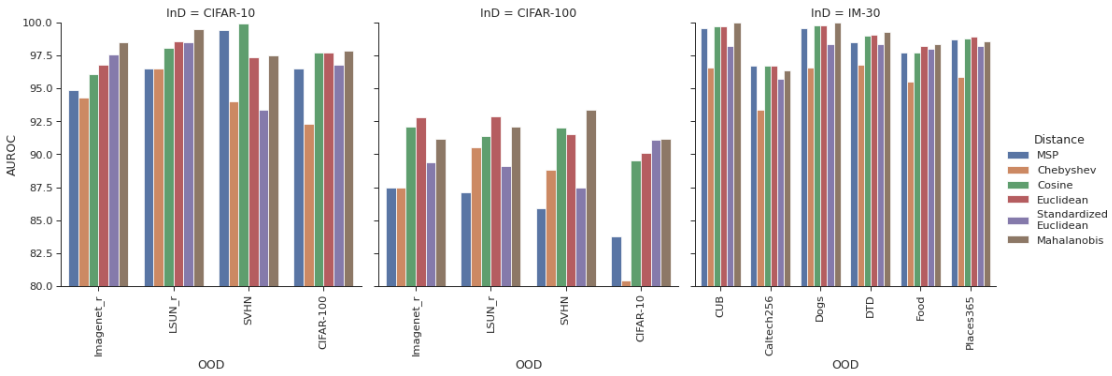
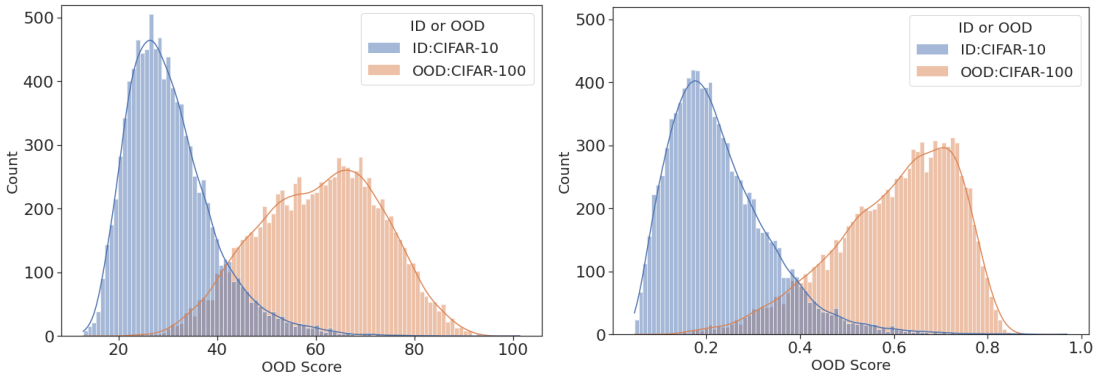


Figure 11. Bar plots of ViT OOD detection AUROC results of selected distance metric. InD dataset separates bar plots. On the x-axis is the OOD dataset, and on the y-axis are the AUROC results. Colour is for the distance metric.

From the results in Table 4, we can see that Mahalanobis distance consistently outperforms (Average Rank 1.8) all other distance metrics in OOD detection. Good results using Mahalanobis distance show evidence that it is still the best distance metric to use in OOD detection when using model's embedding space. Furthermore, we see the similarity of performance of Mahalanobis compared to Cosine distance (+/- 2.5%), which confirms the findings of Koner *et al.* [Kon+21], who stated that they perform similarly—indicating that the angle between InD class means and OOD data contains much information for OOD detection.

Interestingly, Euclidean distances AUROC performance is over 90% for all dataset combinations and +/- 2% from Mahalanobis and has a better average rank (Average Rank



(a) (b)

Figure 12. Histograms of distances from the nearest InD class mean. InD is CIFAR-10 and OOD is CIFAR-100. a) Mahalanobis distance (AUROC 97.9%) b) Cosine distance (AUROC 97.7%).

2.3) than Cosine (Average Rank 2.8). Standardised Euclidean is inconsistent as it has substantially worse performance when the OOD dataset is SVHN (4%). The performance of Standardised Euclidean is worse than Euclidean, suggesting that removing the class variance negatively impacts OOD detection. What is surprising is the performance of Chebyshev (Average Rank 5.6) - although the worst and most inconsistent - still works for OOD detection using ViT— suggesting that ViT constructs a discriminatory embedding that works even if calculating the distance by just taking the axis with the largest difference. MSP, considered a baseline method, works for ViT but yields worse results and is not consistent - performance when CIFAR-100 is InD is consistently worst.

To visually show the delineation of InD and OOD using Mahalanobis and Cosine distance in more detail, see Figure 12. InD samples from CIFAR-10 are blue, and OOD samples from CIFAR-100 are orange. Based on the OOD detection performance of both distances (see Table 4), we expect the distribution of the distances of OOD and InD to be in two distinct groups, and that is evident in the figure as well. The distance of an individual sample for both cases is not 0 because we take the minimum distance between a sample to InD class means. Furthermore, some OOD distances are nearer than InD, which is shown as light purple, which coincides with the result not being 100% AUROC for both cases.

Repeating the same experiments on ResNet50, two significant findings piqued interest. Firstly, Cosine distance OOD detection performance was substantially better compared to Mahalanobis distance. The performance of Cosine distance indicates that the angle between InD and OOD is large. Secondly, MSP yielded in some datasets near similar results to ViT, which indicates that pre-training and upsampling images positively impact OOD detection performance. But it is seen that the performance is not consistent with

all distances indicating that the embeddings are not wholly discriminatory to OOD data, but OOD data has different properties. The results can be found in Appendix I.

In conclusion, Mahalanobis distance is the best distance metric for OOD detection using ViT. Cosine and Euclidean distance are tailing Mahalanobis in performance. Overall, no distance metric failed OOD detection indicating the robustness of various distance metrics and strengthening the claim that ViT builds discriminative embeddings that are useful for OOD detection straight out-of-the-box. Furthermore, ViT is generalisable because OOD detection performance is consistent on all datasets.

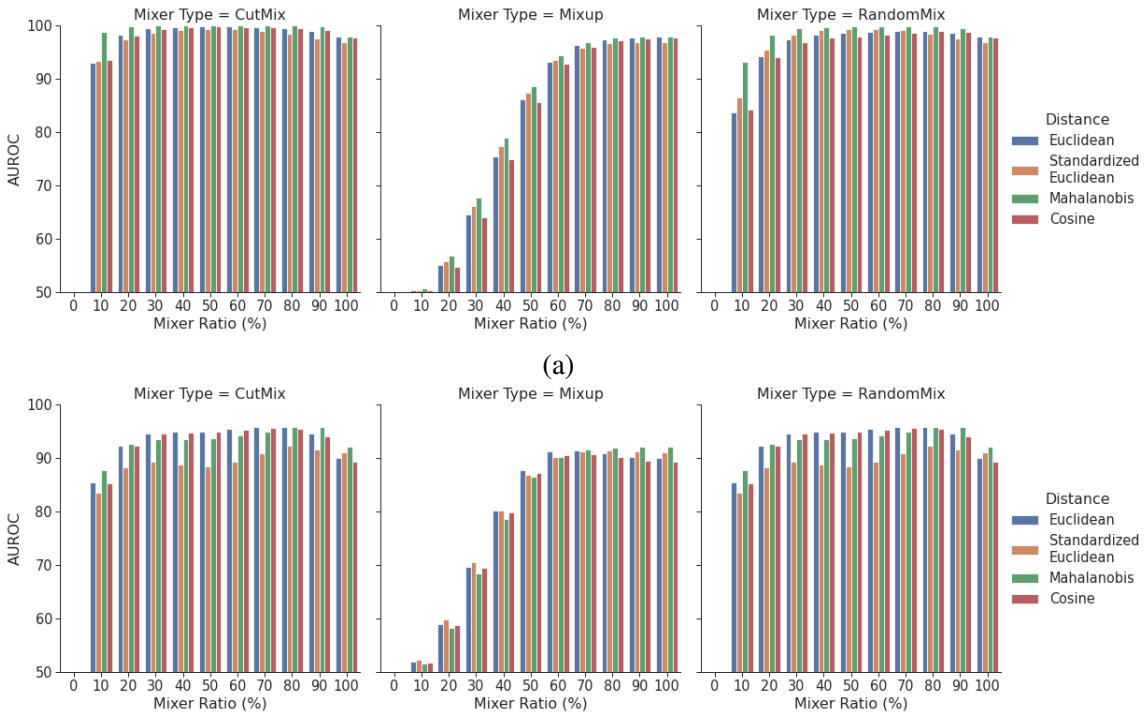
4.3 Out-Of-Distribution Detection Using Mixer Images

We design this experiment to capture the effects of gradual InD to OOD distribution shift, to see how different models and distance metrics act during a gradual distribution shift. We take the InD test set and OOD dataset and mix OOD into InD with increasing mixing ratios. At every ratio, we calculate the OOD detection performance. The goal of the experiment is to achieve two points: to demonstrate the sensitivity of ViTs to OOD detection and subsequently show which distance metric is more sensitive to distribution shift in both cases. A better model should create an embedding that can discriminate OOD at very low mixer ratios with any distance metric. However, a good distance metric should lead to a better OOD detection score at the same mixer ratio compared to other methods. Furthermore, we expect OOD detection performance to increase with the increase in the mixer ratio.

For the experiment, we take one DEIT B-16 fine-tuned on InD CIFAR-10 and a second on InD CIFAR-100. For both models, the far-OOD dataset is SVHN. For near-OOD detection performance, we use CIFAR-100 for the CIFAR-10 fine-tuned model and *vice versa*. In the case of CIFAR-10 to CIFAR-100 and *vice versa*, we want to see how the model distinguishes OOD if we mix two very similar datasets and when it starts to notice the data becoming OOD. For a far-OOD dataset, we use SVHN, which we expect to detect with lower mixer ratios and better OOD detection performance. We use three different mixer logics: RandomMix, CutMix and Mixup, and four distance metrics: Euclidean, Cosine, Standardized Euclidean and Mahalanobis. The description of the experiment on Mixer images and example are in Section 3.5. Near-OOD detection results are summarised in Figure 13 and far-OOD detection results in Figure 14. For ResNet50 results, see Appendix II and Appendix III.

For near-OOD detection, we see that both the CIFAR-10 InD model and CIFAR-100 model have no difficulty detecting RandomMix and CutMix mixing. The OOD detection performance is high for RandomMix and CutMix mixing logic at a 10% mixing ratio. This shows that if outright pixels change on the image, the models have no difficulty detecting it as OOD. However, we see Randommix at a 10% mixer ratio, Mahalanobis distance exhibit better performance than other distance metrics. It shows correlations between the features that play a role in distinguishing between InD and OOD. What is

surprising that in the case of CIFAR-100 Euclidean and Cosine distance exhibit similar or better OOD detection performance between 10-90% Mixer Ratio. Furthermore, what is evident is the fact that for RandomMix and CutMix, the OOD detection performance is higher between mixer ratios of 20-90% compared to when the whole dataset is OOD. It is easier to classify it as OOD because it can distinguish whether the image contains random pixels or a cutout contains another image. Finally, Mahalanobis distance in most cases yields the highest performance or is close to the highest performer irrespective of mixer ratio and logic. In CIFAR-10 to CIFAR-100 with Mixup, Mahalanobis outperforms at every step. Figure 13 summarises the results of the near-OOD detection. The equivalent ResNet results are in Appendix II, and they show similar tendencies. However, Cosine distance performs better than Mahalanobis in all cases, indicating that the angle between mean InD class embedding and OOD embedding is enough for OOD detection using ResNet. The results coincide with the results in the previous section.



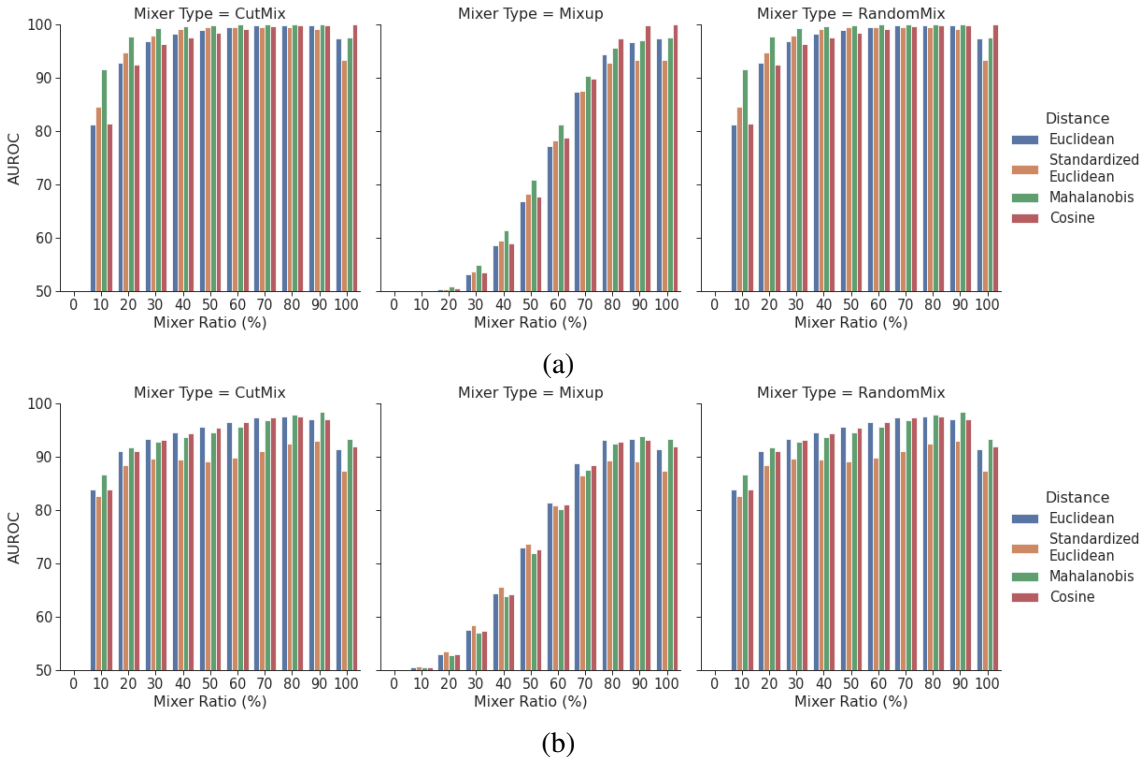
(a)

(b)

Figure 13. Mixer images near-OOD detection AUROC performance with various mixer ratios. a) CIFAR-10 (InD) to CIFAR-100 (OOD), b) CIFAR-100 (InD) to CIFAR-10 (OOD).

For far-OOD detection, we see a repeating pattern regarding how quickly ViT detects RandomMix and CutMix as OOD. So cutting another image or taking random pixels

from OOD data are easily distinguishable as OOD (near the same performance as fully OOD at 20% mixer ratio). No performance metric trumps all with a considerable margin except Mahalanobis at lower mixer ratios. However, when CIFAR-100 is the InD dataset, Standardised Euclidean has the worst results with RandomMix and CutMix when the mixer ratios are between 10% and 90%. Nevertheless, overall, all tested distance metrics work. A significant difference noted between mixer images in near-OOD is that OOD detection performance with Mixup does not reach a plateau at 60% mixer ratio. Adding on to that Cosine distance has a marginal improvement in OOD detection for CIFAR-10 at higher mixer ratios and reaches a near perfect OOD detection score, which coincides with the results in Table 4. Figure 14 summarises the results. The equivalent ResNet results are in Appendix III and mostly show similar tendencies but differ in two major aspects. Firstly, Cutmix is not detected as OOD straight away with lower mixer ratios but increases until it reaches fully OOD. Secondly, the distances are not that similarly performing as with ViT.



(a)

(b)

Figure 14. Mixer images far-OOD detection AUROC performance with various mixer ratios. a) CIFAR-10 (InD) to SVHN (OOD), b) CIFAR-100 (InD) to SVHN (OOD).

Distance gradually increases as we increase the mix ratio of OOD image (CIFAR-100) into ID image (CIFAR-10), as shown in Figure 14. We extracted the model embedding

for each mixer image to construct this visualisation with three different mixer logics and gradually increased the mixer ratio. After which, we calculated the distance to the InD image for each mixer image using either Mahalanobis or Cosine distance. The experiment was repeated 150 times with different InD and OOD image pairs. 0% mixer ratio means that it is the original InD image and 100% is the OOD image. The results show that each image pair took different paths until it entirely became OOD, shown in light grey. The dark grey line is the aggregated line that shows the mean estimate and its 95% confidence. From the results, we see six different aggregated lines. Firstly, Mixup with Mahalanobis OOD detection performance increase plateaus at 70% mixer ratio. For Cosine distance, until 20% of the distance, it does not go further from the InD image, after which it increases up to 80% until reaching a plateau. The results coincide with the mixup results in Figure 13 (a), where OOD detection performance using Mahalanobis is higher than Cosine distance. The reasoning for this is that Mahalanobis distance contains the information of all the class mean embeddings, which are helpful for OOD detection. Meanwhile, the angle between InD and OOD is not that evident at lower mixer ratios, leading to worse performance of OOD detection using cosine distance. For Cutmix and RandomMix mixer logics, the distance increases drastically at earlier mixer ratios when using Mahalanobis distance, again coinciding with the results (see Figure 13 a). The RandomMix behaves similarly in case of Mahalanobis or Cosine distance. However, the distance between InD and OOD increases gradually when using CutMix mixer logic, which is also evident in the OOD detection performance.

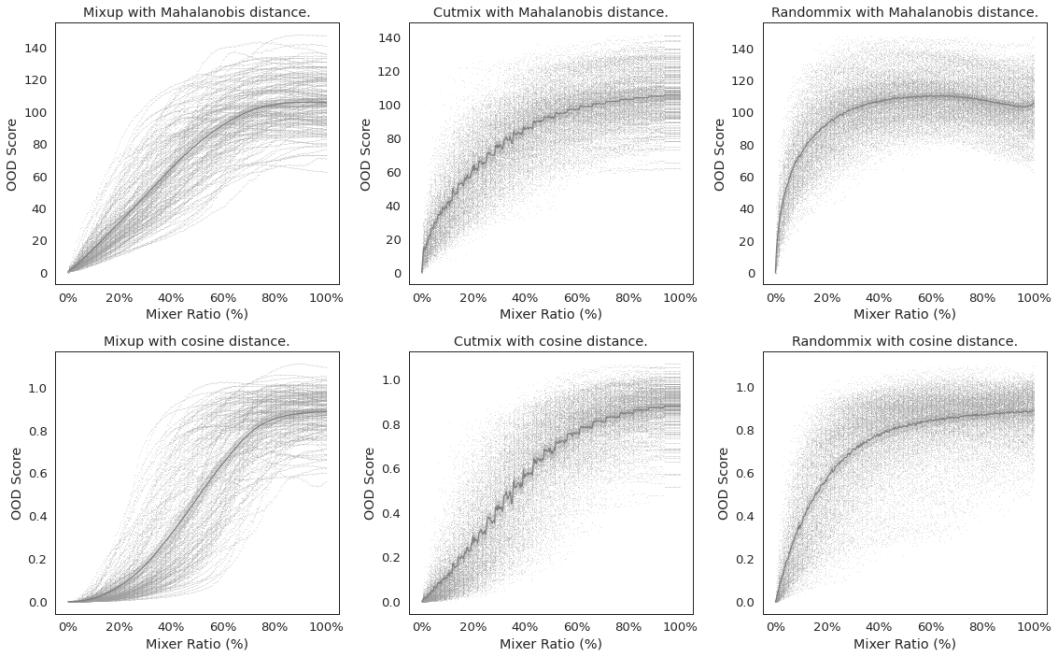


Figure 15. InD image (CIFAR-10) to OOD (CIFAR-100) image embedding distance. x-axis denotes the mixing ratio of how much OOD is in the InD image and y-axis is the distance between the InD image and OOD image.

In conclusion, ViT's react to OOD shift quite quickly - 10% of OOD in an InD with either RandomMix or Cutmix and most of the dataset is considered OOD. On the other hand, Mixup is detected gradually but plateaus when more than half of the image is OOD. In most cases, Mahalanobis distance is the most well-performing distance metric, but other distance metrics behave similarly, indicating that the ViT embedding is discriminatory. Comparing ViT with Resnet, the same tendencies are mostly there. However, the overall performance is worse for near-OOD detection, and the distance metrics do not act similarly, with the best distance metric being Cosine. For far-OOD detection, the difference with ResNet was that CutMix was detected gradually, whereas ViT detected it as OOD lower mixer ratios. Regarding distance metrics for ViT, Mahalanobis seems to be the most sensitive to OOD with lower mixer ratios and, in most cases, has the best OOD detection performance.

5 Conclusions

The main objective of the thesis was to evaluate the performance of a fine-tuned ViT for OOD Detection in multiple scenarios to confirm the claims that SOTA performance can be achieved out-of-the-box without any outlier exposure or specific training methods. Secondly, to confirm the robustness and explain ViT embedding space using various distance metrics. Finally, explore the OOD detection performance of a ViT in a controlled environment. We investigated the use of ViT for OOD detection in the following three experiments.

In the first experiment, we tested the OODformer framework's capabilities for OOD detection using a pre-trained ViT as a feature extractor and using Mahalanobis distance to delineate between InD and OOD compared to a pre-trained ResNet50. We concluded that both pre-trained models perform well on OOD detection, with ViT having a better performance in most InD and OOD combinations. The most significant performance gap between ResNet and ViT was evident in near-OOD detection, the most challenging OOD detection task. Furthermore, ViT was more generalisable by having consistent results across all InD and OOD combinations. Overall, the results confirm the findings of previous works by Koner *et al.* [Kon+21], and Fort *et al.* [FRL21] that ViT can be used for OOD-detection out-of-the-box without any outlier exposure or training methods.

In the second experiment, we tested different distance metrics on the latent embedding space of the ViT. We conducted this experiment to confirm the claim that ViT is robust to various distance metrics and to determine if any distance metric is better for OOD detection using ViT. The results indicated that Mahalanobis distance is the best distance metric for OOD detection using ViT. The results coincide with previous research, which indicated that Mahalanobis distance is highly effective at OOD detection [Lee+18; Kon+21; FRL21]. However, Cosine and Euclidean distances are close to Mahalanobis in performance when using ViT as the feature extractor. Overall no distance metric failed OOD detection indicating the robustness of various distance metrics and strengthening the claim that ViT builds discriminatory embeddings useful for OOD detection. Moreover, ViT is generalisable because OOD detection performance is consistent on all datasets and using any distance metric.

In the third experiment, we used mixed InD and OOD images to capture the effects of gradual InD to OOD distribution shift. When OOD was mixed into InD by replacing random pixels or cutting a part from OOD into InD, the shift was detected quite quickly. On the other hand, when blending InD and OOD images, results showed that increasing the OOD amount in InD is equivalent to increasing the distance between InD and OOD. Blending InD and OOD images increases the OOD detection rate until the performance plateaus when more than half of the image is OOD. In most cases, Mahalanobis distance is the most well-performing distance metric, but other distance metrics behave similarly, indicating that the ViT embedding is discriminatory. In addition, the results coincide with previous experiments that indicated that ViT is robust to distance metrics.

In addition, in the third experiment, we compared ViT with Resnet; the same tendencies are observed there. However, the overall performance of OOD detection is worse for near-OOD detection, and distance metrics in the embedding space do not act similarly. This shows that the latent embedding space of ResNet is not as discriminatory as ViT. Moreover, for far-OOD detection, the difference with ResNet was that cutting OOD into InD was detected gradually, whereas ViT detected this change as OOD with lower mixer ratios.

In conclusion, ViT is capable of OOD detection out-of-the-box without any custom training methods or outlier exposure. ViT has considerably better performance at near-OOD detection and is more robust to various distance metrics than a similarly trained ResNet. When experimenting with ViT OOD detection in a controlled distribution shift, it is evident that pasting random pixels or cutting a random part of an OOD image into an InD image is quickly detected as OOD by ViT. However, blending OOD into InD is equivalent to increasing the distance between InD and OOD. ViT needs at least half of the OOD image to be blended in to have comparable performance to a fully OOD input in these scenarios. Overall the best distance metric to use in the latent space of ViT is Mahalanobis.

The findings of this thesis have to be seen in light of some limitations. We calculate a shared covariance matrix due to numerical stability instead of calculating a covariance matrix for each class for Mahalanobis distance. The numerical instability comes from having fewer samples than features in the embedding in some datasets. Thus, the multiplication of the covariance matrix and their inverses did not result in an identity matrix. Secondly, we only consider DeiT Base-16 in this work as the ViT equivalent and do not investigate the effect of size on OOD detection - although other works show it has a positive effect [Kon+21]. Finally, we do not consider any other training methods in this thesis - meaning that the model's hyper-parameters might not be optimal. Future works can consider the ensembling of different distance metrics to utilise the properties of multiple distance metrics. Furthermore, one can consider applying the same methodology with other ANN architectures.

6 References

[Amo+16] Dario Amodei, Christopher Olah, Jacob Steinhardt, Paul Francis Christiano, John Schulman, and Dandelion Mané. "Concrete Problems in AI Safety". In: *ArXiv* abs/1606.06565 (2016).

[BKH16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. "Layer normalization". In: *arXiv preprint arXiv:1607.06450* (2016).

[Ber+19] Maxim Berman, Hervé Jégou, Andrea Vedaldi, Iasonas Kokkinos, and Matthijs Douze. *MultiGrain: a unified image embedding for classes and instances*. 2019. arXiv: 1902.05509 [cs.CV].

[Blo] Peter Bloem. Accessed on 16.05.2022. URL: http://peterbloem.nl/blog/transformers.

[BGV14] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. "Food-101 – Mining Discriminative Components with Random Forests". In: (2014). Ed. by David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, pp. 446–461.

[Bro20] Jason Brownlee. *What is deep learning?* Accessed on 08.05.2022. Aug. 2020. URL: https://machinelearningmastery.com/what-is-deep-learning/.

[Cho+15] Kittipong Chomboon, Pasapitch Chujai, Pongsakorn Teerarassammee, Kittisak Kerdprasop, and Nittaya Kerdprasop. "An Empirical Study of Distance Metrics for k-Nearest Neighbor Algorithm". In: Jan. 2015, pp. 280–285. DOI: 10.12792/iciae2015.051.

[CLH17] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. "A downsampled variant of imagenet as an alternative to the cifar datasets". In: *arXiv preprint arXiv:1707.08819* (2017).

[Cim+14] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. "Describing Textures in the Wild". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 3606–3613. DOI: 10.1109/CVPR.2014.461.

[Cub+20] Ekin Dogus Cubuk, Barret Zoph, Jon Shlens, and Quoc Le. "RandAugment: Practical Automated Data Augmentation with a Reduced Search Space". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 18613–18624.

[Den+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. IEEE. 2009, pp. 248–255.

[Dev+18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

[Dos+20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *CoRR* abs/2010.11929 (2020). Accessed on 08.05.2022. arXiv: 2010.11929. URL: https://arxiv.org/abs/2010.11929.

[ED20] Rezvan Ehsani and Finn Drabløs. "Robust Distance Measures for kNN Classification of Cancer Data". In: *Cancer Informatics* 19 (2020). PMID: 33116353, p. 1176935120965542. DOI: 10.1177/1176935120965542. URL: https://doi.org/10.1177/1176935120965542.

[FRL21] Stanislav Fort, Jie Ren, and Balaji Lakshminarayanan. "Exploring the limits of out-of-distribution detection". In: *Advances in Neural Information Processing Systems* 34 (2021).

[Gao21] Peter Gao. *The unreasonable effectiveness of neural network embeddings*. Accessed on 15.05.2022. Mar. 2021. URL: https://medium.com/aquarium-learning/the-unreasonable-effectiveness-of-neural-network-embeddings-93891acad097.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org Accessed on 08.05.2022. MIT Press, 2016.

[GHP07] Gregory Griffin, Alex Holub, and Pietro Perona. "Caltech-256 Object Category Dataset". In: (2007). Accessed on 08.05.2022. URL: https://authors.library.caltech.edu/7694/.

[He+16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[HG16a] Dan Hendrycks and Kevin Gimpel. "A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks". In: *CoRR* abs/1610.02136 (2016). arXiv: 1610.02136. URL: http://arxiv.org/abs/1610.02136.

[HG16b] Dan Hendrycks and Kevin Gimpel. "Gaussian error linear units (gelus)". In: *arXiv preprint arXiv:1606.08415* (2016).

[Hen+20] Dan Hendrycks, Xiaoyuan Liu, Eric Wallace, Adam Dziedzic, Rishabh Krishnan, and Dawn Song. "Pretrained Transformers Improve Out-of-Distribution Robustness". In: (2020), pp. 2744–2751. DOI: 10.18653/v1/2020.acl-main.244.

[HMD19] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. "Deep Anomaly Detection with Outlier Exposure". In: *International Conference on Learning Representations*. Accessed on 08.05.2022. 2019. URL: https://openreview.net/forum?id=HyxCxhRcY7.

[Hen+19] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. "Using self-supervised learning can improve model robustness and uncertainty". In: *Advances in Neural Information Processing Systems* 32 (2019).

[HVD+15] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. "Distilling the knowledge in a neural network". In: *arXiv preprint arXiv:1503.02531* 2.7 (2015).

[Hof+20] Elad Hoffer, Tal Ben-Nun, Itay Hubara, Niv Giladi, Torsten Hoefler, and Daniel Soudry. "Augment Your Batch: Improving Generalization Through Instance Repetition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[Hua+16] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. "Deep networks with stochastic depth". In: *European conference on computer vision*. Springer. 2016, pp. 646–661.

[Kho+11] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. "Novel Dataset for Fine-Grained Image Categorization". In: *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*. Colorado Springs, CO, June 2011.

[KK20] H. Kinsley and D. Kukieła. *Neural Networks from Scratch in Python: Building Neural Networks in Raw Python*. Harrison Kinsley, 2020. URL: https://books.google.ee/books?id=Ll1CzgEACAAJ.

[Kon+21] Rajat Koner, Poulami Sinhamahapatra, Karsten Roscher, Stephan Günnemann, and Volker Tresp. "OODformer: Out-Of-Distribution Detection Transformer". In: *CoRR* abs/2107.08976 (2021). Accessed on 08.05.2022. arXiv: 2107.08976. URL: https://arxiv.org/abs/2107.08976.

[KH+09] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images". In: (2009).

[Lee+18] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. "A simple unified framework for detecting out-of-distribution samples and adversarial attacks". In: *Advances in Neural Information Processing Systems* 2018-Decem (LID 2018), pp. 7167–7177. ISSN: 10495258.

[Liu22] Frank Liu. *Understanding neural network embeddings*. Accessed on 08.05.2022. Apr. 2022. URL: https://frankzliu.com/blog/understanding-neural-network-embeddings.

[Net+11] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Ng. "Reading Digits in Natural Images with Unsupervised Feature Learning". In: *NIPS* (Jan. 2011).

[NYC15] Anh Nguyen, Jason Yosinski, and Jeff Clune. "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 427–436.

[PT17] C.R. Parikh and H. Thiessen Philbrook. "Chapter Two - Statistical Considerations in Analysis and Interpretation of Biomarker Studies". In: *Biomarkers of Kidney Disease (Second Edition)*. Ed. by Charles L. Edelstein. Second Edition. Academic Press, 2017, pp. 21–32. ISBN: 978-0-12-803014-1. DOI: https://doi.org/10.1016/B978-0-12-803014-1.00002-9. URL: https://www.sciencedirect.com/science/article/pii/B9780128030141000029.

[Ren+21] Jie Ren, Stanislav Fort, Jeremiah Liu, Abhijit Guha Roy, Shreyas Padhy, and Balaji Lakshminarayanan. "A simple fix to mahalanobis distance for improving near-ood detection". In: *arXiv preprint arXiv:2106.09022* (2021).

[Rud16] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016).

[SR15] Takaya Saito and Marc Rehmsmeier. "The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets". In: *PLOS ONE* 10 (Mar. 2015), pp. 1–21. DOI: 10.1371/journal.pone.0118432. URL: https://doi.org/10.1371/journal.pone.0118432.

[Sch15] Jürgen Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural Networks* 61 (2015), pp. 85–117. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2014.09.003. URL: https://www.sciencedirect.com/science/article/pii/S0893608014002135.

[Smi17] Leslie N. Smith. "Cyclical Learning Rates for Training Neural Networks". In: *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2017, pp. 464–472. DOI: 10.1109/WACV.2017.58.

[Tou+20] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. "Training data-efficient image transformers & distillation through attention". In: *CoRR* abs/2012.12877 (2020). Accessed on 08.05.2022. arXiv: 2012.12877. URL: https://arxiv.org/abs/2012.12877.

[Vas+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[Wah+11] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. "The Caltech-UCSD Birds-200-2011 Dataset". In: (2011). Accessed on 08.05.2022. URL: http://www.vision.caltech.edu/visipedia/CUB-200-2011.html.

[Wig19] Ross Wightman. *PyTorch Image Models*. https://github.com/rwightman/pytorch-image-models. Accessed on 08.05.2022. 2019. DOI: 10.5281/zenodo.4414861.

[Win+20] Jim Winkens, Rudy Bunel, Abhijit Guha Roy, Robert Stanforth, Vivek Natarajan, Joseph R. Ledsam, Patricia MacWilliams, Pushmeet Kohli, Alan Karthikesalingam, Simon Kohl, Taylan Cemgil, S. M. Ali Eslami, and Olaf Ronneberger. "Contrastive Training for Improved Out-of-Distribution Detection". In: (2020), pp. 1–18. URL: http://arxiv.org/abs/2007.05566.

[Yu+15] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. "LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop". In: *CoRR* abs/1506.03365 (2015). arXiv: 1506.03365. URL: http://arxiv.org/abs/1506.03365.

[YA19] Qing Yu and Kiyoharu Aizawa. "Unsupervised out-of-distribution detection by maximum classifier discrepancy". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9518–9526.

[Yun+19] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. "Cutmix: Regularization strategy to train strong classifiers with localizable features". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 6023–6032.

[Zha+21] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. "Dive into Deep Learning". In: *arXiv preprint arXiv:2106.11342* (2021).

[Zha+18] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. *mixup: Beyond Empirical Risk Minimization*. 2018. arXiv: 1710.09412 [cs.LG].

[Zho+20] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. “Random erasing data augmentation”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 07. 2020, pp. 13001–13008.

[Zho+17] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. “Places: A 10 million Image Database for Scene Recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).

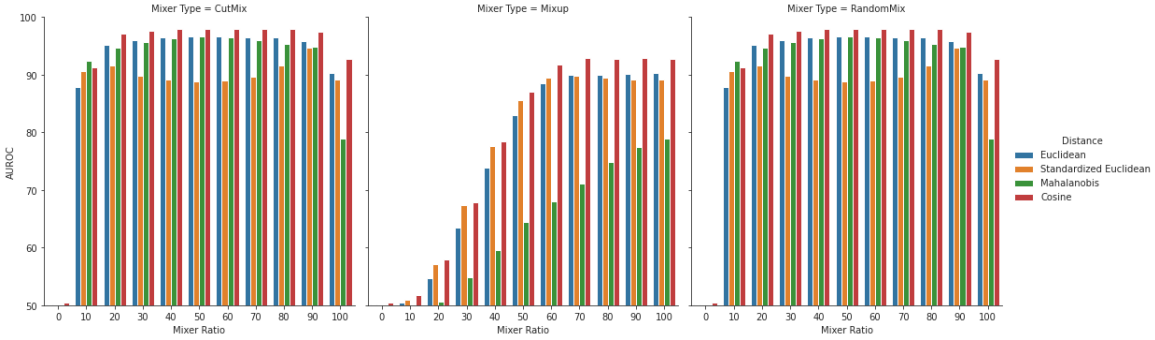
Appendix

I. OOD detection performance AUROC with all considered distances for Resnet50 and ViT

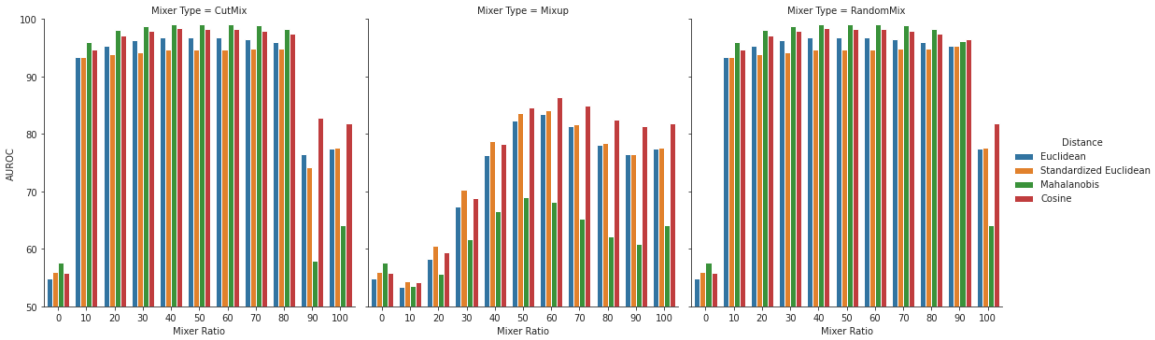
Model	InD	OOD	MSP[a]	CHE[b]	COR[c]	COS[d]	EUC[e]	RBF[f]	SEUC[g]	MAHA[h]
ResNet50	CIFAR-10	Imagenet_r	94.4	84.6	95.1	**95.7**	93.4	93.4	91.8	87.1
		LSUN_r	95.9	87.5	96.7	**97.2**	95.6	95.6	94.1	86.2
		SVHN	97.2	84.7	**98.3**	**98.3**	97.9	97.9	89.1	94.9
		CIFAR-100	92.3	77.6	**92.7**	**92.6**	90.1	90.1	89.4	79.0
	CIFAR-100	Imagenet_r	86.6	76.0	92.3	93.5	91.1	91.1	90.5	**94.0**
		LSUN_r	86.5	73.9	92.7	93.3	91.6	91.6	92.0	**94.2**
		SVHN	85.3	84.2	92.5	94.8	95.3	95.3	94.3	**97.2**
		CIFAR-10	84.1	70.5	**85.2**	84.5	80.8	80.8	81.5	71.3
	IM-30	CUB	**95.6**	83.4	94.0	95.4	92.6	92.6	92.5	79.8
		Caltech-256	**94.4**	79.9	93.9	94.1	90.9	90.9	88.9	76.6
		Dogs	94.9	81.2	**97.0**	96.9	95.2	95.2	92.2	89.5
		DTD	95.5	90.8	97.6	**98.6**	97.6	97.6	94.2	97.7
		Food-101	**88.4**	68.2	83.0	83.3	83.2	83.2	83.5	81.0
		Places-365	**95.4**	79.5	94.6	95.1	91.9	91.9	91.7	75.5
ViT	CIFAR-10	Imagenet_r	94.9	94.3	96.1	96.1	96.8	96.8	97.6	98.5
		LSUN_r	96.5	96.5	98.1	98.1	98.6	98.6	98.5	99.5
		SVHN	99.4	94.0	99.9	99.9	97.4	97.4	93.4	97.5
		CIFAR-100	96.5	92.3	97.7	97.7	97.7	97.7	96.8	97.9
	CIFAR-100	Imagenet_r	87.5	87.5	92.1	92.1	**92.8**	**92.8**	89.4	91.2
		LSUN_r	87.1	90.5	91.4	91.4	92.9	92.9	89.1	**92.1**
		SVHN	85.9	88.8	92.0	92.0	91.5	91.5	87.5	**93.4**
		CIFAR-10	83.8	80.4	89.4	89.5	90.1	90.1	91.1	**91.2**
	IM-30	CUB	99.6	96.6	99.7	99.7	99.7	99.7	98.2	**100.0**
		Caltech-256	**96.7**	93.4	**96.7**	**96.7**	**96.7**	**96.7**	95.7	96.4
		Dogs	99.6	96.6	99.8	99.8	99.8	99.8	98.4	**100.0**
		DTD	98.5	96.8	99.0	99.0	99.1	99.1	98.4	**99.3**
		Food-101	97.7	95.5	97.7	97.7	98.2	98.2	98.0	**98.4**
		Places-365	98.7	95.9	98.8	98.8	**98.9**	**98.9**	98.2	98.6

[a]Maximum Softmax Probability
[b]Chebyshev distance
[c]Correlation distance
[d]Cosine distance
[e]Euclidean distance
[f]RBF Kernel
[g]Standardized Euclidean Distance
[h]Mahalanobis distance

II. Near-OOD detection AUROC mixer images using ResNet50 as the feature extractor



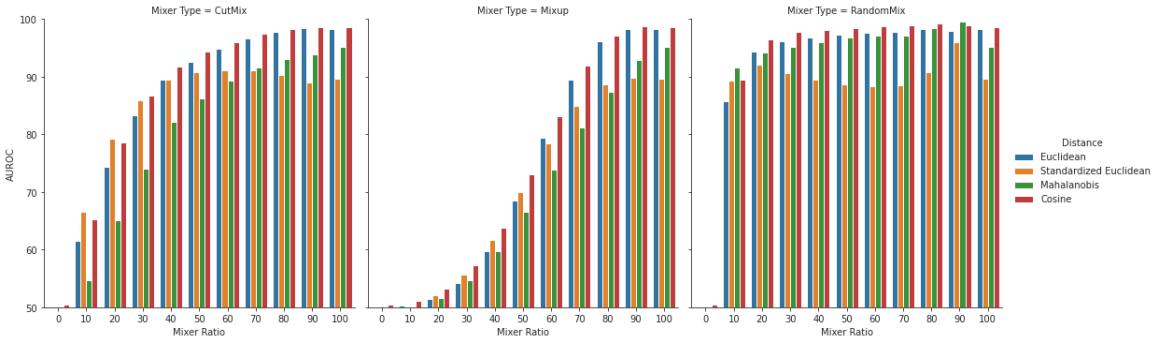
(a)



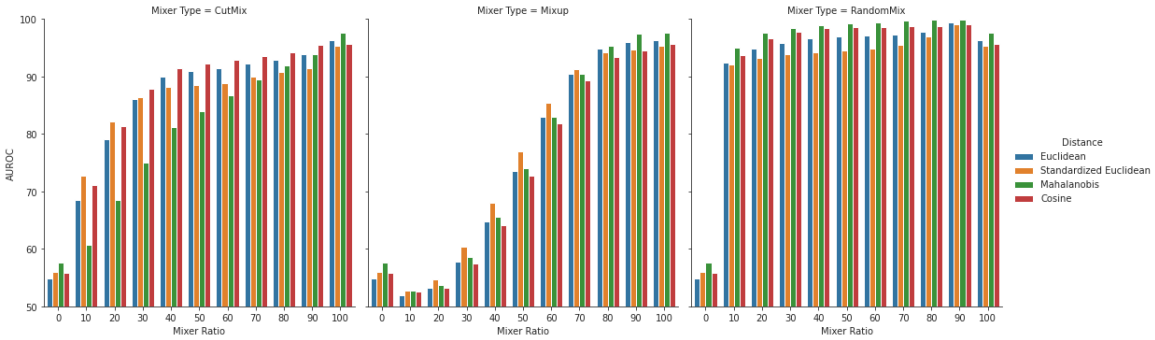
(b)

Mixer images near-OOD detection AUROC performance with various mixer ratios. a) CIFAR-10 to CIFAR-100, b) CIFAR-100 to CIFAR-10.

III. Far-OOD detection AUROC mixer images using ResNet50 as the feature extractor



(a)



(b)

Mixer images far-OOD detection AUROC performance with various mixer ratios. a) CIFAR-10 to SVHN, b) CIFAR-100 to SVHN

IV. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Karl Kaspar Haavel**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Out-Of-Distribution Detection Using Vision Transformers,

   supervised by Meelis Kull and Bhawani Shankar Leelar.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Karl Kaspar Haavel
17/05/2022