# ODIN: OUTLIER DETECTION IN NEURAL NETWORKS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Adoption of deep learning in safety-critical systems raise the need for understanding what deep neural networks do not understand. Several methodologies to estimate model uncertainty have been proposed, but these methodologies constrain either how the neural network is trained or constructed. We present Outlier Detection In Neural networks (ODIN), an assumption-free method for detecting outlier observations during prediction, based on principles widely used in manufacturing process monitoring. By using a linear approximation of the hidden layer manifold, we add prediction-time outlier detection to models after training without altering architecture or training. We demonstrate that ODIN efficiently detect outliers during prediction on Fashion-MNIST, ImageNet-synsets and speech command recognition.

## 1 INTRODUCTION

Thanks to the powerful transformations learned by deep neural networks, deep learning is applied in an increasing number of applications. But deep neural networks, as all data-driven models, tend to fail when input data differs from training data. Adopting deep learning in increasingly complex and possibly safety-critical systems makes it crucial to know not only whether the model's predictions are accurate, but also whether the model should predict at all. If the model is able to detect outlier observations post-training directly, the system can fall back to safe behaviour minimizing negative consequences of faulty predictions. By understanding the limits of models' learned representations and detecting when observations are not recognized, autonomous decision making based on deep learning can be improved.

In manufacturing process control, predictive models have long been used to predict process outcomes as well as detecting outlier input for decades (Eriksson et al., 2013; Kourti et al., 1996; Ferrer, 2014). A widely used model for this purpose is Partial Least Squares regression (PLS) (Wold et al., 2001), which project input data onto a set of linear latent variables prior to prediction. In the latent variable space, the distance from new observations to the training data distribution is used to detect outlier observations. The latent variables can also be used to approximate input observations, meaning that outliers can also be detected by measuring the distance to the latent variable subspace itself. However, the layers of a neural network learn a non-linear mapping from input to output data spaces rather than a single linear subspace. This makes it difficult to directly determine the limits of a neural network model's knowledge in the same manner as for a PLS model.

In this paper we present Outlier Detection In Neural networks (ODIN), a method for detecting outliers during prediction in deep neural networks. Based on principles long used in manufacturing process control, we propose using a linear approximation of intermediate activations to provide a fixed representation of the training data. By comparing new data to this fixed representation we are able to detect outliers during prediction without imposing constraints on architecture or training. This allows us to use ODIN as a plug-in method allowing reliable and safe autonomous decision making based on deep learning.

## 2 RELATED WORK

A wide collection of methods allowing neural networks to describe uncertainty in predictions, which can be used to determine if new observations are outliers, have been proposed. For decades, many methods have been formulated within a Bayesian framework (Denker & LeCun, 1991; MacKay,

1992) allowing neural networks to predict probability distributions rather than point inferences. The predictive uncertainty can then be estimated by the entropy or variance of the predicted distribution. Gal & Ghahramani (2016) proposed MC-dropout, using prediction time dropout (Hinton et al., 2012) and Monte-Carlo sampling. In summary, MC-dropout make multiple predictions per inference while the network is randomly perturbed by drop-out which results in a predicted distribution.

A number of alternatives to using dropout to perturb Monte-Carlo samples have been proposed in recent years including: sampling based on batch-normalization parameters (Teye et al., 2018), model ensembles (Lakshminarayanan et al., 2017), multiple prediction heads in a shared base network (Osband et al., 2016; Ilg et al., 2018; Mandelbaum & Weinshall, 2017), variational inference of weight distribution instead of regular point weights (Blundell et al., 2015) and Laplace approximation of distributions from existing weights (Ritter et al., 2018). However, the mentioned methods constrain either how the network is constructed (Lakshminarayanan et al., 2017; Osband et al., 2016; Ilg et al., 2018; Mandelbaum & Weinshall, 2017) or how the network is trained (Gal & Ghahramani, 2016; Teye et al., 2018) limiting their use in systems already in production. Several methods also rely on multiple inferences per prediction (Gal & Ghahramani, 2016; Teye et al., 2018; Blundell et al., 2015; Ritter et al., 2018). This limits their use in real-time systems or systems with limited computational resources.

An alternative approach for estimating uncertainty in classification problems is presented by Chen et al. (2018) where linear classifiers are trained to classify the target output given intermediate layers of a given base model. The linear classifier outputs are then fed to a meta-model that is trained to estimate whether or not the base model is correct. Another alternative approach is proposed by Lee et al. (2018) that leverage Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) to augment the original dataset with border-line outliers. A deep neural classifier is then trained to output high uncertainty for outlier observations and low uncertainty for the original observations. This method does however involve training a GAN that can be difficult to train to convergence (Mescheder et al., 2018).

Anomaly detection is closely related to prediction time outlier detection, and there are many methods for flagging deviating observations. Non neural-methods include one-class support vector machines (Schölkopf et al., 2001), local observation density (Breunig et al., 2000), distances (Knorr et al., 2000), isolation forests (Liu et al., 2008) and many others. A multitide of methods based on deep neural networks, typically autoencoders have been developed as well (Zhou & Paffenroth, 2017; Chalapathy et al., 2017; Zong et al., 2018; Oh & Yun, 2018). Of particular relevance to this work is Oh & Yun (2018), that use reconstruction residual as metric to flag outliers. Important to note is that outlier detection systems are based on training a separate model to detect deviant observations. Prediction time outlier detection, on the other hand, describes the limits of a predictive model's knowledge.

## 3 Prediction-Time Outlier Detection

In this section we briefly describe the Partial Least Squares regression model, and how its latent variable approximation of the input data space is used to detect outliers after training. We then describe how we can apply similar principles in neural networks by using a linear approximation of the hidden layer manifold, in a method we call Outlier Detection In Neural networks (ODIN).

### 3.1 Outlier detection in Partial Least Squares regression

Partial least squares regression (PLS) (Wold, 1975; Geladi & Kowalski, 1986) is a widely used regression model within manufacturing process control. Similar to Principal Component Analysis (PCA), PLS assumes that high-dimensional data resides in a sub-space of the original data space spanned by so called latent variables and formulated in terms of matrix decomposition. The PLS model is summarized as:

$$\begin{aligned} \boldsymbol{X} &= \boldsymbol{T}\boldsymbol{P}^T + \boldsymbol{E} \\ \boldsymbol{Y} &= \boldsymbol{T}\boldsymbol{C}^T + \boldsymbol{F} \end{aligned} \tag{1}$$

where the $n \times m$ input matrix $\boldsymbol{X}$ is decomposed into $n \times k$ latent variable matrix $\boldsymbol{T} = [\boldsymbol{t}_1 \ ... \ \boldsymbol{t}_k]$ and $m \times k$ loading matrix $\boldsymbol{P} = [\boldsymbol{p}_1 \ ... \ \boldsymbol{p}_k]$ with residual matrix $\boldsymbol{E}$. The $n \times p$ response matrix $\boldsymbol{Y}$ is predicted using $\boldsymbol{T}$ multiplied with response weight matrix $\boldsymbol{C} = [\boldsymbol{c}_1 \ ... \ \boldsymbol{c}_k]$ with residuals $\boldsymbol{F}$. The latent variable-matrix $\boldsymbol{T}$ spans a orthogonal subspace of the data-matrix $\boldsymbol{X}$ and maximize the covariance between $\boldsymbol{X}$ and $\boldsymbol{Y}$. Note that the PLS model of $\boldsymbol{X}$ is similar to how PCA approximates the data through matrix decomposition but PCA finds latent variables $\boldsymbol{t}_i$ that maximize the variance in $\boldsymbol{X}$ rather than the covariance between two matrices.

The columns in $\boldsymbol{T}$ are typically calculated sequentially, where the first column $\boldsymbol{t}_1$ is found through basis-vectors $\boldsymbol{w}_1 \in \mathbb{R}^m$ and $\boldsymbol{c}_1 \in \mathbb{R}^p$ solving the optimization problem:

$$\begin{aligned} \text{maximize} \quad & \text{Cov}(\boldsymbol{X}\boldsymbol{w}_1, \boldsymbol{Y}\boldsymbol{c}_1) = \boldsymbol{w}_1^T \mathbb{E}(\boldsymbol{X}\boldsymbol{Y}^T)\boldsymbol{c}_1 \\ \text{s.t.} \quad & ||\boldsymbol{w}_1|| = 1 \text{ and } ||\boldsymbol{c}_1|| = 1 \end{aligned} \tag{2}$$

The corresponding loading vector $\boldsymbol{p}_1$ is then chosen so that $\tilde{\boldsymbol{X}}_1 = \boldsymbol{X} - \boldsymbol{t}_1\boldsymbol{p}_1^T$ is uncorrelated with $\boldsymbol{t}_1$. This is achieved by selecting $\boldsymbol{p}_1$ as:

$$\boldsymbol{p}_1 = \frac{\boldsymbol{X}^T\boldsymbol{t}_1}{||\boldsymbol{t}_1||_2^2} \tag{3}$$

The subsequent vectors $\boldsymbol{t}_i$, $\boldsymbol{w}_i$, $\boldsymbol{p}_i$ where $i \in [2, ..., k]$ are then calculated by repeating equations 2 and 3 using $\tilde{\boldsymbol{X}}_i = \tilde{\boldsymbol{X}}_{i-1} - \boldsymbol{t}_{i-1}\boldsymbol{p}_{i-1}^T$ instead of $\boldsymbol{X}$.

The latent variable formulation means that PLS carries its own model of the training data and provides two ways of detecting outliers during prediction. Since new observations are projected to the low-dimensional sub-space spanned by the latent variables, both distance to the sub-space itself and distance to training observations within the sub-space can be used to detect outliers. Distance to sub-space is typically measured using the residual sum of squares (RSS). RSS for a new observation row vector $\boldsymbol{x}_{new} \in \mathbb{R}^p$ is given by:

$$\text{RSS} = \sum_{j=1}^{p} (\boldsymbol{x}_{new,j} - \hat{\boldsymbol{x}}_{new,j})^2 \tag{4}$$

where $\boldsymbol{x}_{new}$ is approximated as $\boldsymbol{x}_{new} \approx \hat{\boldsymbol{x}}_{new} = \boldsymbol{t}_{new}\boldsymbol{P}^T = \boldsymbol{x}_{new}\boldsymbol{P}\boldsymbol{P}^T$.

There are several ways to estimate the distance to training observations within the sub-space and a common choice is the Mahalanobis distance. The Mahalanobis distance is a well-used statistical distance measuring how many standard deviations away an observations is from the origin in a multivariate probability normal distribution. Given a fitted PLS model, the training data projections can be approximated as a multivariate normal distribution with covariance matrix $\boldsymbol{C}_T = \mathbb{E}(\boldsymbol{T}^T\boldsymbol{T})$. Then the Mahalanobis distance for $\boldsymbol{x}_{new}$ is given by:

$$d_{new} = \sqrt{\boldsymbol{t}_{new}^T \boldsymbol{C}_T^{-1} \boldsymbol{t}_{new}} \tag{5}$$

Alternatively, to compensate for using a linear model of a possibly non-linear manifold, a density based metric within the latent variable space may be used. For instance, by using the Local Outlier Factor (LOF) (Breunig et al., 2000), observations within low-density regions may be flagged as outliers instead of only using the Mahalanobis distance.

### 3.2 Outlier detection in neural networks (ODIN) using linear manifold approximation

In contrast to PLS, data are not typically linearly mapped to a single sub-space in deep neural networks. Instead, a neural network performs as a nested series of non-linear transformations. That is, the activation vector $\boldsymbol{a}_i$ of an observation vector $\boldsymbol{x}$ from a layer $i$ is given by:

$$\boldsymbol{a}_i = f_i\Big(\boldsymbol{W}_i f_{i-1}\Big(\boldsymbol{W}_{i-1} f_{i-2}\big(... f_1(\boldsymbol{W}_1\boldsymbol{x})\big)\Big)\Big) \tag{6}$$

with weight-matrices $\boldsymbol{W}_k$ and activation functions $f_k$.

According to the manifold hypothesis, data is assumed to reside near a region of low dimensionality that may be highly entangled. One possible explanation for why deep neural networks work well is that they are able to disentangle complicated manifolds (Brahma et al., 2016). If deep neural networks disentangle manifolds, we may find a transformation from a complicated data manifold to a manifold that is approximately linear. If this hypothesis holds true, we can apply a simple trick to add prediction time outlier detection to neural network models by building a model of the data representation within the model. Given $n \times m$ activation matrix $\boldsymbol{A}_i = [\boldsymbol{a}_{i,1}\ ...\ \boldsymbol{a}_{i,n}]^T$ of training data $\boldsymbol{X}$, where the row vectors $\boldsymbol{a}_{i,k}$ are given by equation 6, we approximate the activation manifold using PCA as:

$$\boldsymbol{A}_i \approx \boldsymbol{T}_{A_i}\boldsymbol{P}_{A_i}^T \tag{7}$$

where the $n \times k$ latent variable matrix $\boldsymbol{T}_{A_i}$ contain the projections of the $\boldsymbol{A}_i$ onto the orthonormal sub-space spanned by columns of the $m \times k$ loading matrix $\boldsymbol{P}_{A_i}$.

Now we have a fixed orthogonal approximation of the activation manifold that we can use to detect outliers during prediction analogously to PLS (see 3.1). Meaning that we can measure the distance from new observations to the activation manifold as the residual sum of squares similar to equation 4 using the observation activation $\boldsymbol{a}_{i,new}$ with projection $\boldsymbol{t}_{\boldsymbol{a}_{i,new}} = \boldsymbol{a}_{i,new}\boldsymbol{P}_{A_i}$:

$$\text{RSS}_{\boldsymbol{a}_{i,new}} = \sum_{j=1}^{m}\Big((\boldsymbol{a}_{i,new} - \boldsymbol{t}_{\boldsymbol{a}_{i,new}}\boldsymbol{P}_{A_i}^T)_j\Big)^2 \tag{8}$$

Similarily, the distance from training observations within the manifold can be measured using Mahalanobis distance or density based approaches as the Local Outlier Factor within the linear approximation. For the Mahalanobis distance, the covariance matrix of the activation projections $\boldsymbol{C}_{T_{A_i}} = \mathbb{E}(\boldsymbol{T}_{A_i}^T\boldsymbol{T}_{A_i})$ is used in in equation 5 as:

$$d_{\boldsymbol{a}_{i,new}} = \sqrt{\boldsymbol{t}_{\boldsymbol{a}_{i,new}}^T\boldsymbol{C}_{T_{A_i}}^{-1}\boldsymbol{t}_{\boldsymbol{a}_{i,new}}} \tag{9}$$

We choose to call our method Outlier Detection In Neural networks, ODIN, since we use the intermediate data representations within the neural network itself. In contrast to common Bayesian approaches, we do not perturb predictions to produce prediction distributions. We simply measure the distance from new observations to the training data to determine whether they are outliers or not.

## 4 EXPERIMENTS

In the following sections we demonstrate how to detect outliers during prediction time using ODIN on different classification tasks. We choose classification tasks for demonstration since it is straightforward to simulate outliers by excluding a subset of the classes. We also explore how to choose what layer's activations to use and rank of PCA approximation. For comparison, we also perform outlier detection using MC-Dropout (Gal & Ghahramani, 2016) since it is well-established and straightforward to implement even though it has received criticism (Osband et al., 2016).

### 4.1 FASHION-MNIST

To provide a simple classification problem with outliers encountered during prediction, we use the Fashion-MNIST (Xiao et al., 2017) dataset. Fashion-MNIST consists of 70 000 greyscale 28x28 pixel images, out of which 10 000 are test set images, of ten categories of fashion products. We excluded five classes to use as outliers, including all shoes (sandals, ankle boots and sneakers) and two clothing classes (pullover and shirts). The intuition is that shoe-images are strong outliers

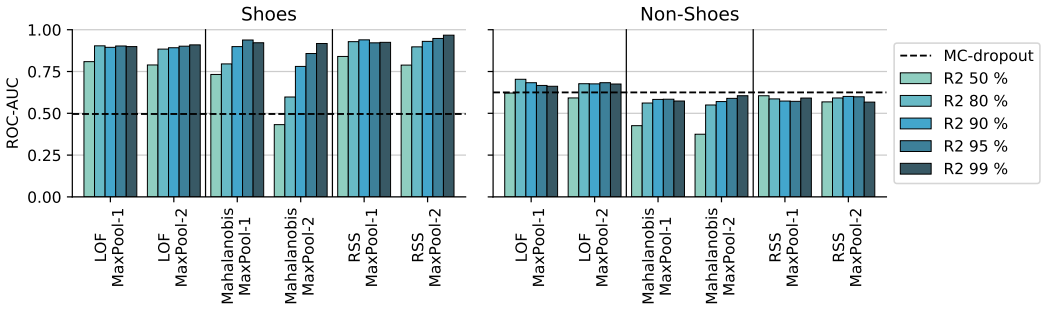


Figure 1: ROC-AUC of prediction time outlier detection on Fashion-MNIST detecting strong outliers represented by shoes (left) and subtle outliers represented as excluded non-shoe garments (right). Results from MC-dropout are shown as horizontal dashed lines.

since all shoe-related information is absent from training data, and excluded clothes are more subtle outliers since the training data contain other upper body garments. We trained a small convolutional neural network (CNN, for architecture see Figure A.1) on five out of the ten classes. We used rmsprop (Tieleman & Hinton, 2012) optimization, categorical cross entropy loss function, batch size 128 for 10 epochs and kept 10 % of the images as validation set and achieved a test set accuracy of 97 %. To use for outlier detection, we extracted features from both max-pooling layers (without global average pooling) for all images.

We evaluate ODIN using different outlier metrics (RSS, Mahalanobis distance and LOF), five levels of explained variance (R2) of the PCA model (50-99 %) and different layers of extracted features using the area under the receiver operating characteristic curve (ROC-AUC) as performance metric, (see Figure 1, complete results in Table A.1). We calculate the ROC-AUC comparing how well test set observations are separated from outlier observations. For comparison, we also used MC-dropout to calculate the image-wise entropy from 50 Monte Carlo samples per image and evaluated the results using ROC-AUC in the same way as ODIN.

All metrics clearly separate strong outliers (shoes) from the test set images (Figure 1 left) with RSS being most sucessful (ROC-AUC 0.97 compared to Mahalanobis 0.94 and LOF 0.91). There is a trend that they benefit from increased PCA R2. Surprisingly MC-dropout failed to detect shoe outliers (ROC-AUC 0.495). The subtle outliers (non-shoes) are significantly more difficult to detect (Figure 1 right), and LOF is most successful doing so (best ROC-AUC 0.71 compared to RSS 0.60, Mahalanobis 0.61 and MC-Dropout 0.63).

To conclude, the Fashion-MNIST experiment show that ODIN successfully detect outliers in a simple image classification problem. Strong outliers seem to be best detected by measuring distance to manifold while subtle outliers are better detected in low-density regions of the linear approximation using LOF.

### 4.2 Cats and Dogs

In order to provide a more complex example we demonstrate prediction time outlier detection using a pre-trained CNN on image synsets from ImageNet (Deng et al., 2009). We train a cat vs. dog classifier on the cat- and dog-synsets from ImageNet and used the car- and horse-synsets as outliers. We used an Inception v3-network (Szegedy et al., 2016) pre-trained on ImageNet, freezing all Inception module weights during training. We replaced the penultimate layer with a hidden layer of 128 ReLu units and a single sigmoid output with 50 % dropout before and after the ReLu-layer and trained for 50 epochs using the Adam optimizer (Kingma & Ba, 2014) and achieved a test set accuracy of 93 %.

We extracted features from each inception module in the Inception-v3 network and pooled them feature-map wise using global average pooling. For each layer of features, we performed outlier detection with five levels of explained variance (R2) for the PCA model (50-99 %) and different outlier

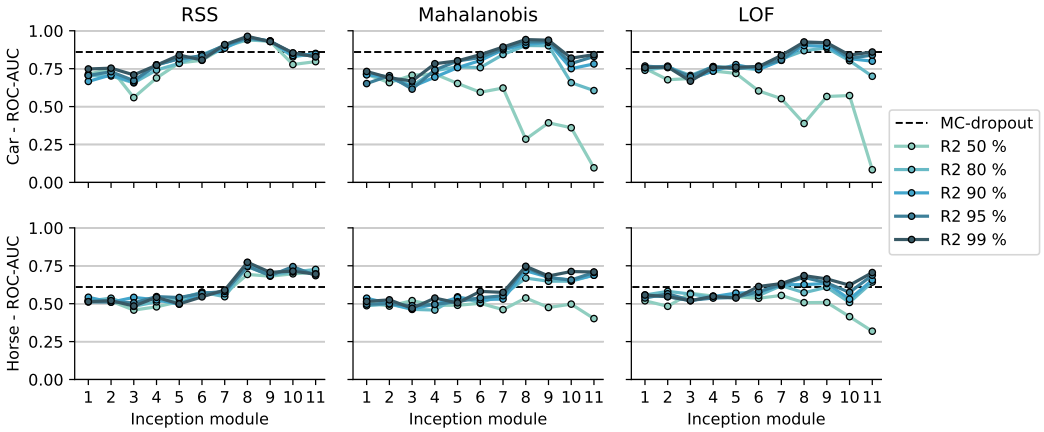


Figure 2: ROC-AUC of prediction time outlier detection on Cats and Dogs experiment detecting strong outliers represented by cars (top row) and subtle outliers represented as horses (bottom row). Inception module, i.e. network depth, of extracted features used for outlier detection is shown on the X-axes. Results from MC-dropout are shown as horizontal dashed lines.

metrics. We evaluated the performance using ROC-AUC in the same manner as in the Fashion-MNIST experiment (see Figure 2, for complete results see tables B.1, B.2 and B.3).

We are able to convincingly detect cars using our cats vs. dogs classifier (best ROC-AUC for RSS, 0.96, Mahalanobis distance 0.94 and LOF 0.93). Horses are also detected as outliers even though they share visual features with cats and dogs (best ROC-AUC for RSS 0.76, Mahalanobis distance 0.75 and LOF 0.69). Since we used dropout for the last fully connected layers, we also performed MC-dropout achieving similar results (ROC-AUC: 0.86 for cars, 0.61 for horses). The degree of explained variance was not as influental in this experiment as in the Fashion-MNIST experiment, but both Mahalanobis distance and LOF fail to detect both cars and horses using 50 % R2. Interestingly, the performance of all metrics peak at inception module 8 where an auxilliary output was used during training on ImageNet (Szegedy et al., 2016).

To conclude, the experiment on cats and dogs show that ODIN reliably detect outliers using a pre-trained CNN on real-world images. ODIN performs slightly better than MC-dropout but does not rely on using dropout, or any type of constraint on the training procedure. In line with the results from the Fashion-MNIST experiment, higher PCA R2 produce more reliable results.

### 4.3 Speech Commands

To show that ODIN for prediction time outlier detection works for not only CNN-based image classification, we perform a speech command recognition experiment using a LSTM-based model. We use the Speech Commands dataset (Warden, 2018) that consists of 105 000 short utterances of 35 words recorded at 16 kHz sampling-rate. The words includes digits *zero* to *nine*, command words *yes*, *no*, *up*, *down*, *left*, *right*, *on*, *off*, *stop*, *go*, *backward*, *forward*, *follow*, *learn* and *visual*. The dataset also include a set of arbitrary words *bed*, *bird*, *cat*, *dog*, *happy*, *house*, *marvin*, *sheila*, *tree* and *wow*. In our experiment, we train a classification model of both digits and command words and use the arbitrary words as outliers.

We transform the utterances into 64 Mel-Frecuency Cepstral Coefficients (Davis & Mermelstein, 1990), using a frame-length of 2048 samples and frame-stride of 512 samples. We train a three layer bi-directional LSTM-model with 30 % dropout after each LSTM-layer and softmax output (see architecture in Figure C.1) for 30 epochs, using the Adam-optimizer (Kingma & Ba, 2014) and batch-size 512 resulting in test-set accuracy of 78 % for the 25 classes. The classification accuracy is lower than the 88 % accuracy of the baseline CNN:s (Warden, 2018), but we believe it is sufficient for demonstrating prediction time outlier detection.

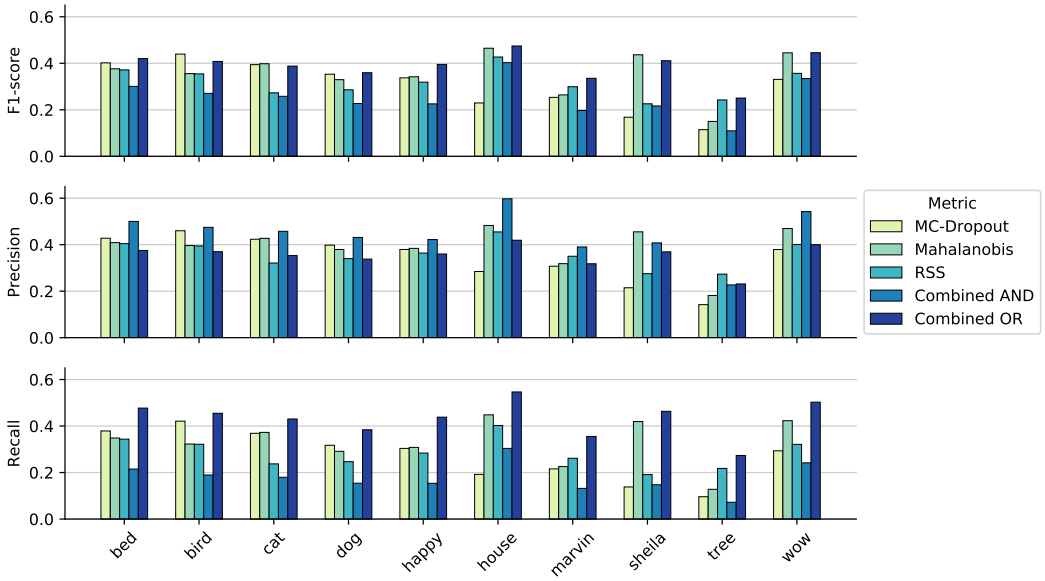


Figure 3: F1-score (top), precision (middle) and recall (bottom) of outlier detection for the words used as outliers in the Speech Commands experiment comparing MC-Dropout and ODIN. Scores for AND- or OR-combining RSS and Mahal classifications are included to show how they influence the precision and recall ratio.

For outlier detection, we extracted training set features from the third LSTM-layer and fitted a PCA-model explaining 99 % of the variance and chose RSS and Mahalanobis distance limits to be the 9th deciles of the training set distances. We then extracted features from the test set and outlier classes and projected them onto the PCA-model and calculated RSS and Mahalanobis distances. Using precision, recall and F1-score we evaluated outlier detection at the 9th deciles (see figure 3, complete results in Table C.1). We also combined RSS and Mahalanobis distance classifications using OR and AND combined classification. For comparison, we also used MC-dropout with 10 Monte Carlo samples per utterance and calculated the sample-wise Shannon entropy. We performed outlier detection using the 9th decile of training set entropies as threshold, and evaluated MC-dropout in the same manner as ODIN.

Detecting outliers in the speech-commands dataset is difficult for both ODIN and MC-dropout with best word-wise F1-scores ranging from 0.25 for *tree*, which is phonetically similar to *three*, to 0.47 for *house*. ODIN consistently outperform MC-dropout. Additionally, since two metrics are used, we also have the opportunity to raise precision or recall by using AND- or OR-combination of classification according to the two metrics. Depending on the application, either precision or recall may be more important than the other.

The Speech Command experiment shows that ODIN performs well for recurrent neural networks on a speech recognition task in addition to image classification. We also demonstrate how it can be used in practice, by selecting classification threshold and evaluating our choice using precision and recall. We also show how combinations of the different metrics available may be used to tune the precision/recall ratio.

## 5 Conclusions

Deep neural networks are powerful transformers that have shown great success in many applications. But, in order to adopt deep learning in safety-critical applications it is crucial to understand when new observations do not match the data used during training. To imitate linear latent variable models used in manufacturing process monitoring, we use a linear approximation of the hidden layer manifolds to measure distance to and within the manifold. We compare our results to MC-dropout, a well

established Bayesian approach, and consistently detect outliers post-training without imposing any constraints on either architecture or training procedure. We demonstrate our method in two image classification experiments, with and without a pre-trained network, and a speech recognition example using a recurrent neural network. By defining the limits of our neural networks' knowledge, ODIN contribute to safer use of deep learning.

## REFERENCES

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight Uncertainty in Neural Network. In *International Conference on Machine Learning*, pp. 1613–1622, 2015.

Pratik Prabhanjan Brahma, Dapeng Wu, and Yiyuan She. Why Deep Learning Works: A Manifold Disentanglement Perspective. *IEEE Trans. Neural Netw. Learning Syst.*, 27(10):1997–2008, 2016.

Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. LOF: Identifying Density-based Local Outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pp. 93–104, 2000.

Raghavendra Chalapathy, Aditya Krishna Menon, and Sanjay Chawla. Robust, Deep and Inductive Anomaly Detection. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 36–51. Springer, 2017.

Tongfei Chen, Jiří Navrátil, Vijay Iyengar, and Karthikeyan Shanmugam. Confidence Scoring Using Whitebox Meta-models with Linear Classifier Probes. *arXiv preprint arXiv:1805.05396*, 2018.

Steven B Davis and Paul Mermelstein. Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. In *Readings in speech recognition*, pp. 65–74. Elsevier, 1990.

Jia Deng, Wei Dong, Richard Socher, Li-jia Li, Kai Li, and Li Fei-fei. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition*, 2009.

John S Denker and Yann LeCun. Transforming Neural-Net Output Levels to Probability Distributions. In *Advances in neural information processing systems*, pp. 853–859, 1991.

Lennart Eriksson, Tamara Byrne, E Johansson, Johan Trygg, and C Vikström. *Multi-and Megavariate Data Analysis: Basic Principles and Applications*, volume 1. Umetrics Academy, 2013.

Alberto Ferrer. Latent Structures-Based Multivariate Statistical Process Control: A Paradigm Shift. *Quality Engineering*, 26(1):72–91, 2014.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *International Conference on Machine Learning*, pp. 1050–1059, 2016.

Paul Geladi and Bruce R Kowalski. Partial Least-Squares Regression: A Tutorial. *Analytica chimica acta*, 185:1–17, 1986.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Eddy Ilg, Özgün Çiçek, Silvio Galesso, Aaron Klein, Osama Makansi, Frank Hutter, and Thomas Brox. Uncertainty Estimates for Optical Flow with Multi-Hypotheses Networks. In *15th European Conference on Computer Vision*, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Edwin M. Knorr, Raymond T. Ng, and Vladimir Tucakov. Distance-Based Outliers: Algorithms and Applications. *The VLDB Journal*, 8(3-4):237–253, February 2000. doi: 10.1007/s007780050006.

Theodora Kourti, Jennifer Lee, and John F Macgregor. Experiences with Industrial Applications of Projection Methods for Multivariate Statistical Process Control. *Computers & chemical engineering*, 20:S745–S750, 1996.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In *Advances in Neural Information Processing Systems*, pp. 6405–6416, 2017.

Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training Confidence-Calibrated Classifiers for Detecting Out-of-Distribution Samples. In *International Conference on Learning Representations*, 2018.

F. T. Liu, K. M. Ting, and Z. H. Zhou. Isolation Forest. In *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422, December 2008. doi: 10.1109/ICDM.2008.17.

David JC MacKay. A Practical Bayesian Framework for Backpropagation Networks. *Neural computation*, 4(3):448–472, 1992.

Amit Mandelbaum and Daphna Weinshall. Distance-based Confidence Score for Neural Network Classifiers. *arXiv preprint arXiv:1709.09844*, 2017.

Lars M. Mescheder, Andreas Geiger, and Sebastian Nowozin. Which Training Methods for GANs do actually Converge? In *35th International Conference on Machine Learning*, 2018.

Dong Yul Oh and Il Dong Yun. Residual Error Based Anomaly Detection Using Auto-Encoder in SMD Machine Sound. *Sensors (Basel, Switzerland)*, 18(5), 2018.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep Exploration via Bootstrapped DQN. In *Advances in Neural Information Processing Systems*, pp. 4026–4034, 2016.

Hippolyt Ritter, Aleksandar Botev, and David Barber. A Scalable Laplace Approximation for Neural Networks. In *Sixth International Conference on Learning Representations*, 2018.

Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. Estimating the Support of a High-Dimensional Distribution. *Neural computation*, 13(7):1443–1471, 2001.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

Mattias Teye, Hossein Azizpour, and Kevin Smith. Bayesian Uncertainty Estimation for Batch Normalized Deep Networks. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Pete Warden. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. *arXiv preprint arXiv:1804.03209*, 2018.

Herman Wold. Path Models with Latent Variables: The NIPALS Approach. In *Quantitative sociology*, pp. 307–357. Elsevier, 1975.

Svante Wold, Michael Sjöström, and Lennart Eriksson. PLS-Regression: A Basic Tool of Chemometrics. *Chemometrics and intelligent laboratory systems*, 58(2):109–130, 2001.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv:1708.07747 [cs, stat]*, August 2017. URL `http://arxiv.org/abs/1708.07747`. arXiv: 1708.07747.

Chong Zhou and Randy C. Paffenroth. Anomaly Detection with Robust Deep Autoencoders. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pp. 665–674, 2017. doi: 10.1145/3097983.3098052.

Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. Deep Autoencoding Gaussian Mixture Model for Unsupervised Anomaly Detection. In *Sixth International Conference on Learning Representations*, 2018.

# APPENDIX A APPENDIX

## A.1 FASHION-MNIST

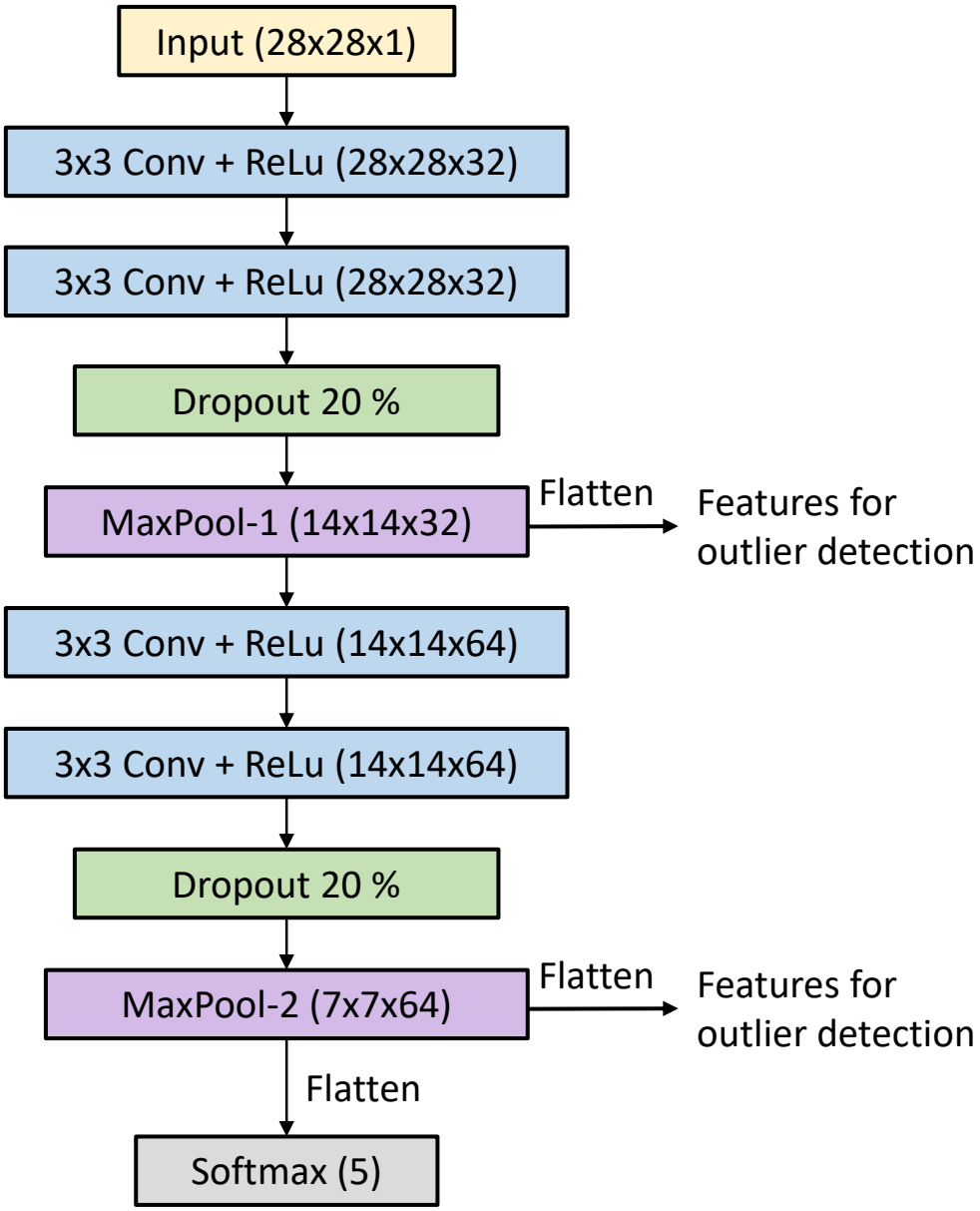


Figure A.1: Architecture of the CNN-model used in the Fashion-MNIST experiments. Numbers in parentheses indicate output dimensions of the current layers.

Table A.1: ROC-AUC for all configurations used in Fashion-MNIST experiment.

| | | | Non-Shoes | Shoes |
|---|---|---|---|---|
| Measure | Features | PCA R2 | | |
| LOF | MaxPool-1 | 0.50 | 0.632584 | 0.806461 |
| | | 0.80 | 0.705666 | 0.904777 |
| | | 0.90 | 0.685191 | 0.894012 |
| | | 0.95 | 0.672836 | 0.905983 |
| | | 0.99 | 0.666276 | 0.900635 |
| | MaxPool-2 | 0.50 | 0.591984 | 0.783379 |
| | | 0.80 | 0.685097 | 0.878966 |
| | | 0.90 | 0.679432 | 0.890894 |
| | | 0.95 | 0.683418 | 0.898365 |
| | | 0.99 | 0.679545 | 0.912277 |
| Mahalanobis | MaxPool-1 | 0.50 | 0.429212 | 0.725880 |
| | | 0.80 | 0.555856 | 0.804085 |
| | | 0.90 | 0.584215 | 0.893431 |
| | | 0.95 | 0.576945 | 0.940009 |
| | | 0.99 | 0.578156 | 0.919574 |
| | MaxPool-2 | 0.50 | 0.382974 | 0.427928 |
| | | 0.80 | 0.543957 | 0.590699 |
| | | 0.90 | 0.577311 | 0.780444 |
| | | 0.95 | 0.589486 | 0.850849 |
| | | 0.99 | 0.605115 | 0.916437 |
| RSS | MaxPool-1 | 0.50 | 0.602458 | 0.847860 |
| | | 0.80 | 0.591201 | 0.927887 |
| | | 0.90 | 0.571679 | 0.936623 |
| | | 0.95 | 0.574044 | 0.918533 |
| | | 0.99 | 0.587556 | 0.923430 |
| | MaxPool-2 | 0.50 | 0.564947 | 0.794195 |
| | | 0.80 | 0.589945 | 0.898555 |
| | | 0.90 | 0.603539 | 0.928036 |
| | | 0.95 | 0.603329 | 0.947959 |
| | | 0.99 | 0.569872 | 0.966362 |
| MC-dropout | - | - | 0.626128 | 0.493604 |

## APPENDIX B CATS AND DOGS

Table B.1: ROC-AUC for all RSS configurations used in Cats and dogs experiment.

| PCA R2 | Inception module | Car | Horse |
|---|---|---|---|
| 0.50 | 1 | 0.709278 | 0.497676 |
| | 2 | 0.746097 | 0.504473 |
| | 3 | 0.576762 | 0.470429 |
| | 4 | 0.699465 | 0.505362 |
| | 5 | 0.789033 | 0.517313 |
| | 6 | 0.793035 | 0.540525 |
| | 7 | 0.887417 | 0.538365 |
| | 8 | 0.934970 | 0.719343 |
| | 9 | 0.931100 | 0.693248 |
| | 10 | 0.781649 | 0.703906 |
| | 11 | 0.794187 | 0.725780 |
| 0.80 | 1 | 0.705079 | 0.568177 |
| | 2 | 0.710353 | 0.516260 |
| | 3 | 0.637867 | 0.526424 |
| | 4 | 0.729584 | 0.535119 |
| | 5 | 0.778540 | 0.518596 |
| | 6 | 0.848603 | 0.540108 |
| | 7 | 0.906682 | 0.567508 |
| | 8 | 0.943456 | 0.758081 |
| | 9 | 0.931845 | 0.697327 |
| | 10 | 0.821543 | 0.715034 |
| | 11 | 0.867583 | 0.715177 |
| 0.90 | 1 | 0.681670 | 0.536555 |
| | 2 | 0.702568 | 0.511546 |
| | 3 | 0.682821 | 0.527521 |
| | 4 | 0.781556 | 0.516929 |
| | 5 | 0.773486 | 0.520306 |
| | 6 | 0.820797 | 0.561280 |
| | 7 | 0.889807 | 0.584810 |
| | 8 | 0.958050 | 0.750493 |
| | 9 | 0.933917 | 0.679214 |
| | 10 | 0.839283 | 0.712403 |
| | 11 | 0.854041 | 0.701439 |
| 0.95 | 1 | 0.698412 | 0.545075 |
| | 2 | 0.741141 | 0.526304 |
| | 3 | 0.691768 | 0.486097 |
| | 4 | 0.775744 | 0.509923 |
| | 5 | 0.802838 | 0.540941 |
| | 6 | 0.837737 | 0.579207 |
| | 7 | 0.905673 | 0.563418 |
| | 8 | 0.960155 | 0.753794 |
| | 9 | 0.933413 | 0.685189 |
| | 10 | 0.848899 | 0.729979 |
| | 11 | 0.836027 | 0.701428 |
| 0.99 | 1 | 0.755548 | 0.549899 |
| | 2 | 0.783529 | 0.524484 |
| | 3 | 0.726799 | 0.488213 |
| | 4 | 0.783420 | 0.499200 |
| | 5 | 0.829163 | 0.502960 |
| | 6 | 0.812442 | 0.559076 |
| | 7 | 0.908622 | 0.565414 |
| | 8 | 0.964234 | 0.739167 |
| | 9 | 0.928095 | 0.691285 |
| | 10 | 0.853077 | 0.715210 |
| | 11 | 0.827277 | 0.701735 |
| MC-dropout | - | 0.874272 | 0.618843 |

Table B.2: ROC-AUC for all LOF configurations used in Cats and dogs experiment.

| PCA R2 | Inception module | Car | Horse |
|---|---|---|---|
| 0.50 | 1 | 0.747062 | 0.506228 |
| | 2 | 0.712447 | 0.487221 |
| | 3 | 0.669598 | 0.552026 |
| | 4 | 0.728203 | 0.551993 |
| | 5 | 0.699202 | 0.533573 |
| | 6 | 0.603274 | 0.536457 |
| | 7 | 0.567212 | 0.563133 |
| | 8 | 0.383306 | 0.535602 |
| | 9 | 0.591717 | 0.508432 |
| | 10 | 0.571455 | 0.400476 |
| | 11 | 0.077551 | 0.317552 |
| 0.80 | 1 | 0.763223 | 0.551675 |
| | 2 | 0.744682 | 0.576301 |
| | 3 | 0.688884 | 0.548649 |
| | 4 | 0.754682 | 0.512598 |
| | 5 | 0.747763 | 0.531227 |
| | 6 | 0.730615 | 0.581553 |
| | 7 | 0.801094 | 0.624731 |
| | 8 | 0.869052 | 0.611969 |
| | 9 | 0.883470 | 0.625948 |
| | 10 | 0.804274 | 0.518387 |
| | 11 | 0.689432 | 0.634227 |
| 0.90 | 1 | 0.780525 | 0.564537 |
| | 2 | 0.764583 | 0.547487 |
| | 3 | 0.711975 | 0.522324 |
| | 4 | 0.733740 | 0.536643 |
| | 5 | 0.767390 | 0.552443 |
| | 6 | 0.764265 | 0.557794 |
| | 7 | 0.820841 | 0.618635 |
| | 8 | 0.902954 | 0.635213 |
| | 9 | 0.916429 | 0.634161 |
| | 10 | 0.805897 | 0.554186 |
| | 11 | 0.799998 | 0.676703 |
| 0.95 | 1 | 0.747972 | 0.579920 |
| | 2 | 0.757642 | 0.549691 |
| | 3 | 0.675628 | 0.514725 |
| | 4 | 0.741327 | 0.556291 |
| | 5 | 0.774473 | 0.529374 |
| | 6 | 0.748180 | 0.580589 |
| | 7 | 0.815929 | 0.623394 |
| | 8 | 0.915607 | 0.654467 |
| | 9 | 0.908326 | 0.652526 |
| | 10 | 0.815633 | 0.567475 |
| | 11 | 0.836487 | 0.684016 |
| 0.99 | 1 | 0.746557 | 0.587880 |
| | 2 | 0.764177 | 0.591805 |
| | 3 | 0.699465 | 0.553276 |
| | 4 | 0.755077 | 0.535602 |
| | 5 | 0.758136 | 0.537180 |
| | 6 | 0.773310 | 0.582332 |
| | 7 | 0.826850 | 0.636759 |
| | 8 | 0.923139 | 0.684981 |
| | 9 | 0.926922 | 0.674828 |
| | 10 | 0.846399 | 0.610927 |
| | 11 | 0.864984 | 0.703895 |
| MC-dropout | - | 0.874272 | 0.618843 |

Table B.3: ROC-AUC for all Mahalanobis distance configurations used in Cats and dogs experiment.

| PCA R2 | Inception module | Car | Horse |
|---|---|---|---|
| 0.50 | 1 | 0.708335 | 0.541325 |
| | 2 | 0.616837 | 0.470078 |
| | 3 | 0.718214 | 0.559318 |
| | 4 | 0.705265 | 0.498235 |
| | 5 | 0.645487 | 0.496645 |
| | 6 | 0.571521 | 0.534713 |
| | 7 | 0.553561 | 0.486930 |
| | 8 | 0.261918 | 0.527005 |
| | 9 | 0.390027 | 0.471438 |
| | 10 | 0.341882 | 0.479771 |
| | 11 | 0.082321 | 0.443434 |
| 0.80 | 1 | 0.702853 | 0.506118 |
| | 2 | 0.692360 | 0.500746 |
| | 3 | 0.636694 | 0.448763 |
| | 4 | 0.738049 | 0.471821 |
| | 5 | 0.763015 | 0.545338 |
| | 6 | 0.751699 | 0.487402 |
| | 7 | 0.850522 | 0.507993 |
| | 8 | 0.914905 | 0.640597 |
| | 9 | 0.900169 | 0.632538 |
| | 10 | 0.662997 | 0.630082 |
| | 11 | 0.626464 | 0.695660 |
| 0.90 | 1 | 0.702995 | 0.518343 |
| | 2 | 0.710210 | 0.517258 |
| | 3 | 0.610357 | 0.459399 |
| | 4 | 0.711855 | 0.533957 |
| | 5 | 0.759704 | 0.506984 |
| | 6 | 0.807125 | 0.517784 |
| | 7 | 0.882330 | 0.551248 |
| | 8 | 0.912471 | 0.718576 |
| | 9 | 0.920168 | 0.670431 |
| | 10 | 0.710473 | 0.635235 |
| | 11 | 0.777926 | 0.689542 |
| 0.95 | 1 | 0.686494 | 0.533299 |
| | 2 | 0.721098 | 0.532510 |
| | 3 | 0.617374 | 0.508837 |
| | 4 | 0.743926 | 0.505614 |
| | 5 | 0.795448 | 0.555107 |
| | 6 | 0.815403 | 0.526479 |
| | 7 | 0.879336 | 0.553616 |
| | 8 | 0.929740 | 0.733433 |
| | 9 | 0.928424 | 0.684038 |
| | 10 | 0.781117 | 0.693994 |
| | 11 | 0.821159 | 0.693193 |
| 0.99 | 1 | 0.702316 | 0.536051 |
| | 2 | 0.710956 | 0.535755 |
| | 3 | 0.653425 | 0.507094 |
| | 4 | 0.793847 | 0.522784 |
| | 5 | 0.796423 | 0.527992 |
| | 6 | 0.841915 | 0.568659 |
| | 7 | 0.889687 | 0.595533 |
| | 8 | 0.939224 | 0.724782 |
| | 9 | 0.931264 | 0.669642 |
| | 10 | 0.833341 | 0.700463 |
| | 11 | 0.844568 | 0.704048 |
| MC-dropout | - | 0.874272 | 0.618843 |

## APPENDIX C SPEECH COMMANDS

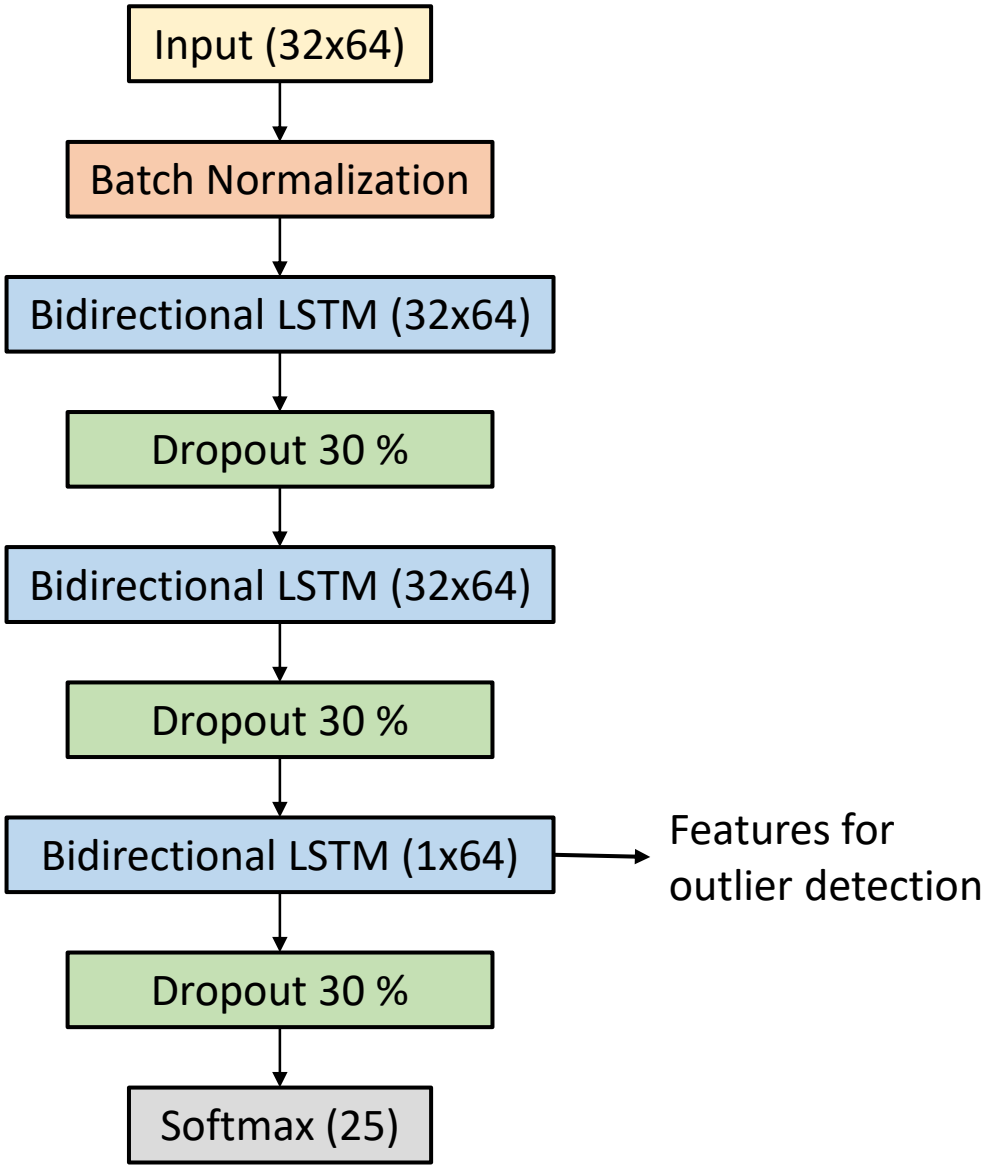


Figure C.1: Architecture of the LSTM-model used in the Speech Commands experiments. Numbers in parentheses indicate output dimensions of the current layer.

Table C.1: F1-score, precision and recall for all outlier words in the Speech Commands experiment.

| Method | Word | F1-score | Precision | Recall |
|---|---|---|---|---|
| MC-dropout | bed | 0.401790 | 0.427691 | 0.378848 |
| | bird | 0.439555 | 0.459788 | 0.421027 |
| | cat | 0.394107 | 0.423164 | 0.368784 |
| | dog | 0.353033 | 0.397995 | 0.317199 |
| | happy | 0.337388 | 0.379331 | 0.303797 |
| | house | 0.229379 | 0.284513 | 0.192144 |
| | marvin | 0.253497 | 0.307327 | 0.215714 |
| | sheila | 0.167971 | 0.214615 | 0.137982 |
| | tree | 0.114615 | 0.142017 | 0.096077 |
| | wow | 0.330767 | 0.378954 | 0.293453 |
| Combined_AND | bed | 0.300694 | 0.500000 | 0.214995 |
| | bird | 0.270776 | 0.474515 | 0.189438 |
| | cat | 0.258042 | 0.457393 | 0.179714 |
| | dog | 0.227068 | 0.431012 | 0.154135 |
| | happy | 0.225473 | 0.421896 | 0.153846 |
| | house | 0.402760 | 0.597209 | 0.303833 |
| | marvin | 0.197153 | 0.390141 | 0.131905 |
| | sheila | 0.216491 | 0.407661 | 0.147379 |
| | tree | 0.109530 | 0.226786 | 0.072200 |
| | wow | 0.334311 | 0.542283 | 0.241639 |
| Combined_OR | bed | 0.420017 | 0.375098 | 0.477160 |
| | bird | 0.407906 | 0.369685 | 0.454942 |
| | cat | 0.387927 | 0.353131 | 0.430330 |
| | dog | 0.359437 | 0.337883 | 0.383929 |
| | happy | 0.395170 | 0.359856 | 0.438169 |
| | house | 0.474430 | 0.419086 | 0.546616 |
| | marvin | 0.335507 | 0.317853 | 0.355238 |
| | sheila | 0.410965 | 0.369188 | 0.463403 |
| | tree | 0.250456 | 0.231028 | 0.273451 |
| | wow | 0.445418 | 0.399925 | 0.502591 |
| Mahalanobis distance | bed | 0.376307 | 0.408853 | 0.348560 |
| | bird | 0.355674 | 0.396193 | 0.322674 |
| | cat | 0.398107 | 0.427201 | 0.372723 |
| | dog | 0.329524 | 0.379205 | 0.291353 |
| | happy | 0.341977 | 0.384102 | 0.308179 |
| | house | 0.464785 | 0.482671 | 0.448178 |
| | marvin | 0.264140 | 0.318334 | 0.225714 |
| | sheila | 0.436551 | 0.455180 | 0.419387 |
| | tree | 0.150050 | 0.181452 | 0.127914 |
| | wow | 0.444995 | 0.469420 | 0.422986 |
| RSS | bed | 0.371544 | 0.404442 | 0.343595 |
| | bird | 0.354417 | 0.394534 | 0.321705 |
| | cat | 0.272933 | 0.321119 | 0.237322 |
| | dog | 0.285948 | 0.340026 | 0.246711 |
| | happy | 0.318928 | 0.363920 | 0.283836 |
| | house | 0.426921 | 0.454789 | 0.402272 |
| | marvin | 0.299346 | 0.350128 | 0.261429 |
| | sheila | 0.225788 | 0.275249 | 0.191395 |
| | tree | 0.242328 | 0.273181 | 0.217737 |
| | wow | 0.356695 | 0.400941 | 0.321244 |