Comparison of Feedforward and Convoluted Neural Networks

Jakob Brown

Abstract— This paper presents a critical comparison and evaluation of two neural networks – namely the feedforward multi-layer perceptron (MLP), and the convolutional neural network (CNN) – in their ability to classify images. Parameters and hyperparameters were varied using a gridsearch-based approach and best-performing networks were selected. For this particular problem, the CNN algorithm performed best on the holdout test set, in terms of classificaiton accuracy as well as F1 score. Results are examined in further detail and explanations for disparity in model performance are highglighted and discussed.

1. Introduction

The applications of computer vision and image classification are myriad, spanning visual search, facial recognition, medical imagery, and more [4]. The Cifar-10 data set [9] is one of the most commonly used sets in the field of training machine learning and computer vision algorithms. Key to the progression of the sophistication and capability of image classification neural networks is the evaluation and comparison of different methods.

Thus, the present paper seeks to contribute to the body of research which has seen convolutional neural networks (CNN) revolutionise the machine capacity to classify images [3][6], by comparing to the simpler multi-layer perceptron (MLP), using the benchmark Cifar 10 dataset.

The paper will be structured as such: First, the two algorithms being compared will be introduced and detailed to conclude this introduction. Then, information on the Cifar-10 dataset will be provided alongside some summary statistics. Following that, methods for selection of neural network architecture for both algorithms will be discussed, as well as hyperparameter choices and justifications, supported by the previous literature. Results generated from the best performing models from each algorithm will be reported in detail, and finally, critical evaluation and reasoning of the results will be carried out.

1.1 Multilayer perceptron (MLP)

Consisting of an input layer, and output layer and at least one fully connected hidden layer, the MLP has a feedforward phase, and a backward phase: In the forward phase, the inputs are ran through the initially randomly assigned weights of nodes of the network, modifying them to produce an output. In the backward phase, the MLP utilises a backpropagation algorithm in combination with an assigned error function to calculate the gradient of the error signal produced with respect to the weights of the nodes of the network and adjusts the weights accordingly [5].

1.2 Convolutional Neural Network (CNN)

CNNs also make use of backpropagation. By sliding a convolving filter across the input data, the CNN differs from the MLP's fully connected layers in that nodes are only connected to a few nodes in the previous layer, reducing computation, as well as preventing overfitting by having too many parameters or weights. The filter convolves (takes the dot product between the data and the filter's weights) over each 'location' of the data, producing one value for each, resulting a smaller map of the data, a feature/activation map. Convolutional layers are made up of multiple feature maps. A CNN typically has several convolutional layers, before passing the features to a fully connected layer. As before with the MLP, a non-linearity function is then applied, and output values are produced, which are then backpropagated through the same error process, using the chain rule to derive the gradients for the parameters of each of the convolutional layers and adjusting the filter's weights accordingly [4].

2. Dataset

The Cifar 10 dataset was chosen to satisfy the specific purposes of the present report's brief, which were to train, run, adapt, test, and evaluate two types of neural networks. It was therefore preferred to use a dataset that was a) not too easy, and would therefore need 'breaking', b) would expect to produce results that differed between algorithms, so as to have a performance disparity to explore and critically evaluate in order to demonstrate knowledge, c) was already relatively clean and ready to be used, so as not to waste unnecessary time on the data pre-processing stage of the technical side of the assignment, and finally d) had sufficient empirical literature making use of the same dataset in order to have sources of inspiration from previous efforts for hyperparameter/architecture/methodology choices. The Cifar 10 dataset satisfied all of these criteria. This data set was obtained as a dataset object that can be loaded straight from the '*torchvision.datasets*' module.

The data set itself consists of 60,000 images, shaped 32 x 32 pixels, coloured by 3 RGB channels. There are 10 classes: Airplane; automobile; bird; cat; deer; dog; frog; horse; ship; truck [9]. An example of the images are shown in *figure 1* below.

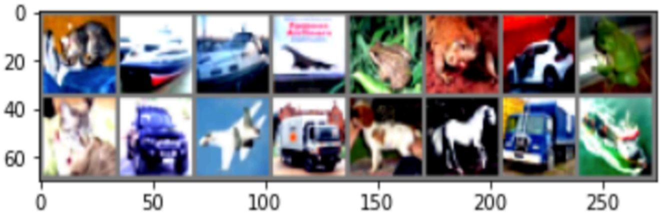

Figure 1. From top left to bottom right: cat, ship, ship, airplane, frog, frog, car, frog, cat, automobile, airplane, truck, dog, horse, truck, ship.

The data is pre-split into a training set of 50,000, and a test set of 10,000. For both sets, the distributions of classes are balanced evenly, e.g. there are 5,000 of each class in the training set and 1,000 of each class in the test set. This saves the need for a visualisation of class distribution, and also adds the benefit of simplifying choice of evaluation metrics, as classification accuracy will not be reflecting the bias of over-selection of class majority by the networks, as it tends to be when there is a significant class imbalance [11].

3. Methods

Given the data set is already pre-split in to training and testing sets, the only choice to make in that regard was with validation set. K-fold cross validation was opted for, in order to give a more robust idea of how well the trained networks would generalise to unseen data by averaging the multiple folds' validation scores, and to be able to modulate the number of folds if necessary, to reduce computational expense [2]. For most of training, a K value of 3 was used, as a balanced trade-off between reliability of validation measures and computational workload/time expense.

The model selection stage was conducted in the form of multiple grid-searches, one for each network across 3 differing number of hidden layers for each, totalling 6. After defining the architecture of the networks, they would be input in to a skorch *NeuralNetClassifier* object, specifying cross-validation as well as call-back functions, including an early stopping criterion. This criterion, and model selection in general was initially based on the validation loss metric, given the tendency for neural networks - especially deep neural networks - to overfit the data and thus generalise poorly to unseen test data [7]. However, it was later discovered that training the networks for longer resulted in best test results despite validation loss plateauing and subsequently increasing. Although this indicates overfitting to the given validation set, it is likely that the networks trained in the present study benefit from the data's consistency in features across train and test set. So, although the bias is high, as is reflected by the rising validation loss (*figure 2*), the predictability of the network remains high too, as the images the network is becoming biased to are so similar to the unseen images [3].

The grid-searches themselves differed across hyperparameters that were available to each network and were deemed relevant and influential to the dataset at hand through exploratory network training and through examination of the literature surrounding the MLP and CNN for image classification (*table 1 & 2*) [1][3][4][7][8][14]. This preliminary process also revealed some parameters that were clearly either uninfluential, or were best left stable as one particular value, such as the optimizer used for each algorithm (stochastic-gradient descent (SGD) for MLP, Adam for CNN: SGD produced vastly poorer results than Adam for CNN, and vice versa for the MLP, and so did not need to be compared in the grid-search). Once the grid-search algorithms were ran, the best performing network in terms of validation accuracy was selected and used to predict on the hold-out test set. Classification reports, as well as visualisations of key metrics were created to aid the comparison of the performance of the two neural network algorithms.

The ReLU activation function, *y = max(0, x),* was employed for both network algorithms, reflecting previous literature on image classification both for MLPs and CNNs [15][4]. The loss function used for both was cross entropy loss, the widely accepted loss function for multi-label classification [8].

3.1 MLP Architecture and Hyperparameters

The MLP hyperparameters that were varied across the grid-searches are shown below in *table 1*. Architecture was kept relatively stable: from the input layer, there was a repeated fully connected layer, activation layer, and dropout layer. As more hidden layers were added, so too were repeated activation and dropout layers in the same order, before the final layer reduces the number of neurons down to 10, the number of classes.

Hyperparameter	Values
Hidden layer 1	1000, 1500, 2000
Hidden layer 2*	500, 750, 1000
Hidden layer 3*	250, 500
Momentum	0.7, 0.9, 0.99
Dropout	0, 0.2, 0.5
Learning rate (lr)	0.0001, 0.001, 0.01, 0.1

Table 1. Hyperparameters varied in grid-search of largest MLP architecture
** These parameters were only varied in the grid-searches that included these layers in the network architecture*

3.2 CNN Architecture and Hyperparameters

The CNN architecture also made use of ReLU activation functions between each convolutional layer. Moreover, max pooling layers combined and selected the most dominant features identified by each convolution to map, as well as reduced the resolution of the images by 2 each layer to make the computations more efficient. Batch normalization was employed to prevent drastic internal covariate shift.

As can be noticed, less hyperparameters tested for the CNN grid-searches, purely due to computation time, which for the 6 layered CNN could take as long as circa 15020 seconds (4.17 hours) per combination. It was therefore necessary to whittle down hyperparameters to be varied and keep some stable based on experimentation with training singular networks or with grid-search results from the smaller CNNs, such as fixing dropout at the optimum value of 0.2.

Hyperparameter	Values
Hidden convolutional layer 1	32, 64
Stride for convolutional layer 4*	1, 2
Hidden fully connected layer 1	1500, 1000
Learning rate (lr)	0.0001, 0.001, 0.01, 0.1

Table 2. Hyperparameters varied in grid-search of largest CNN architecture
** this hyperparameter was only included when the architecture included a 4th convolutional layer.*

4. Results/Algorithm comparison

4.1 Validation

With regards to the network selection, the 2- and 3-layer MLPs both produced similar validation accuracy (*table 3*). This gives credence to the notion that there is no theoretical benefit to using more than 2 hidden layers in an MLP [7]. However, the 3-layer MLP took over twice the time to train, and so the 2-layer architecture was selected as the optimal network to carry forward to the test stage. For the CNN, the greater the number of convolutional layers, the better the validation performance (*table 4*), which aligns with the success of the best performing CNNs to date on the present data set [6]. Stride was best kept uniform, in spite of some pervious research finding a wider stride in the middle of a CNN to be beneficial for extracting different scopes of features of an image [14]. The smallest, most cautious learning rate was found to be the optimum over both algorithms, while a small probability value for unit dropout (0.2) tended to be best at preventing units from becoming dependent on surrounding units. Initial comparison of validation results showed a significant margin of performance between the 2 algorithms (*figure* 2). MLP performance benefitted from large and sparse hidden layers converging into smaller ones before output. The opposite was found to be true for the CNNs, which saw vast performance improvement using the doubling of neurons with each layer, coupled with the dimensionality reduction and regularization techniques of pooling and batch normalization. The highlighted networks from each table represent the one that was selected for testing.

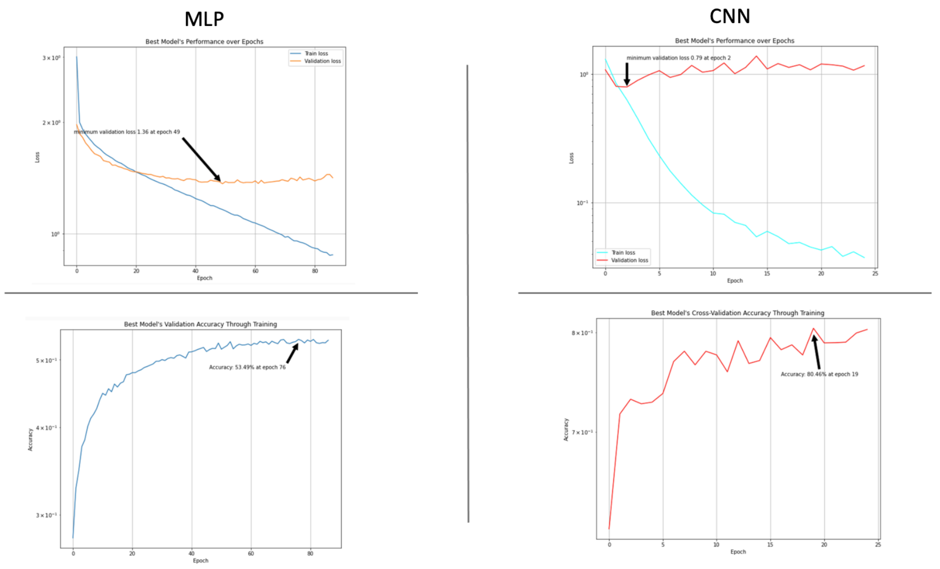

Figure 2. Accuracy and Loss for both training and validation plotted on a semi-log graph for each algorithm.

Network architecture	FC layer 1 nodes	FC layer 2 nodes	FC layer 3 nodes	Dropout	Learning Rate	Momentum	Validation Accuracy	Training time (seconds)
1 hidden layer	1500	N/A	N/A	0.5	0.0001	0.7	44.3	983
2 hidden layers	1500	500	N/A	0.2	0.0001	0.9	53.49	1929
3 hidden layers	2000	1000	500	0.2	0.0001	0.7	54.57	4057

Table 3. Hyperparameters and validation results of the best performing MLP from each of the 3 different proposed network architectures.

Network Architecture	Conv layer 1 nodes	Conv layer 2 nodes	Conv layer 3 nodes	Conv layer 4 nodes	Conv layer 5 nodes	Conv layer 6 nodes	FC layer 1	Conv layer 3 stride	Dropout	Learning Rate	Weight decay	Validation Accuracy	Training time (seconds)
2 conv layers	32	64	N/A	N/A	N/A	N/A	1000	1	0.2	0.0001	0	68.34	3400
4 conv layers	32	64	128	256	N/A	N/A	1000	2	0.2	0.0001	0	74.75	1929
6 conv layers	32	64	128	256	512	1024	1000	1	0.2	0.0001	0	80.46	56291

Table 4. Hyperparameters and validation results of the best performing CNN from each of the 3 different proposed network architectures.

4.2 Test

On the test set, the MLP performed with 53.03% accuracy, while the CNN classified with 80.12% accuracy, showing perfect generalisation to unseen data in spite of the training beyond optimum validation loss. This is likely due to the balance of the data, meaning no classification bias can be learnt by the neural network towards majority classes, no matter how many epochs are trained (*figure* 2) [11].

This is likely to also be a result of cross-validation averaging out any effects of order of min-batches in training the networks.

Confusion matrix visualisation shows that the only significant confusions in classes for the CNN was between the dog and cat classes. This gives an indication of the level of detail of features that had been captured by the CNN, virtually eliminating confusion of shape, size, colour etc. Other than the cat class (0.65), F1 scores for all other classes were above 0.7. Conversely, the MLP appeared to struggle more to discern at this more base-feature level, significantly confusing automobile and truck, airplane and ship etc. This highlights the advantage the convolving filters have over fully connected layers in distinguishing features efficiently by focussing only on pixels of an image that are close together and therefore relate to each other in a meaningful way, rather than flatten the images dimensions into a linear array straight away [8]. Further layering of the convolutional process or a greater number of kernels would likely allow the CNN to define the requisite nuanced features that distinguish a cat and a dog [6].

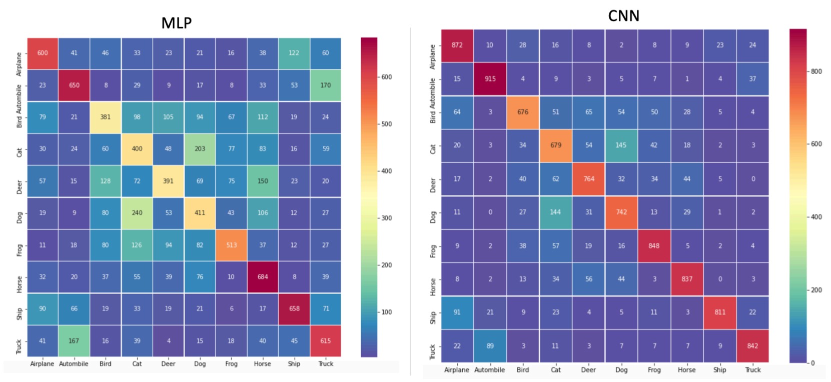

Figure 3. Confusion Matrices for both algorithms on test set.

A different visualisation of test results in the form of ROC curve and precision-recall curve plots further elucidate a stronger set of classification predictions made by the CNN (*figure 4*). Greater average AUC (0.98 to MLP's 0.89) corroborates the stronger sensitivity and specificity scores by the CNN across all classes. Furthermore, the precision-recall curve plots show the variation across classes, for which there is some for both algorithms. It highlights the difficulty had with the cat class in particular, perhaps reflecting the diversity of appearance that cats can have, not just in their physical features, but also in position and angle. Recall score suffered particularly for the MLP with this class, meaning that the number of cat images identified in relation to the total number of cats was poor (0.38). This suggests that no real meaningful features could be discerned that allowed for the consistent recognition of cat images when they were input to the network.

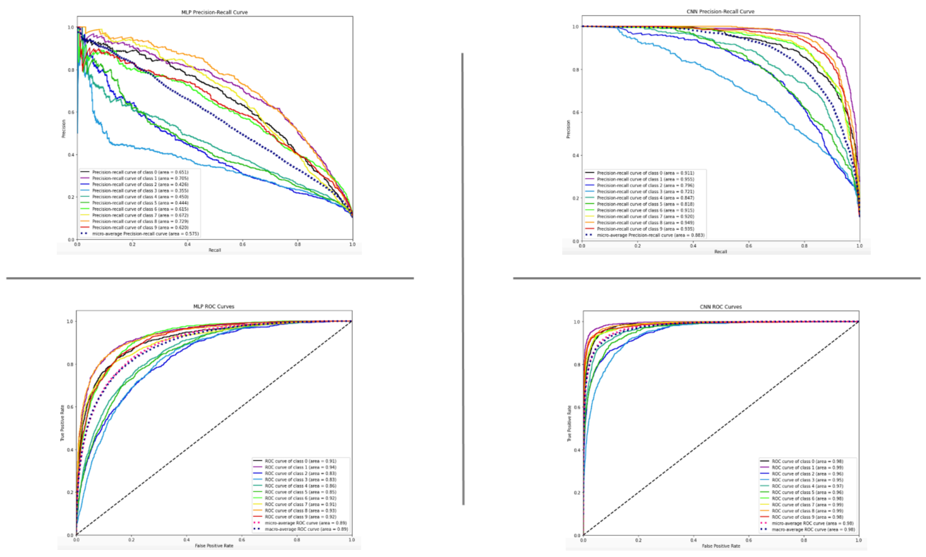

Figure 4. Precision-recall (top) and ROC curve (bottom) plotting for both algorithms on test set. Classes key: 0 = airplane, 1 = Automobile, 2 = bird, 3 = cat, 4 = deer, 5 = dog, 6 = frog, 7 = horse, 8 = ship, 9 = truck.

5. Conclusion

As expected, CNNs outperform MLPs when it comes to image classification, aligning with previous literature[8][12][16]. Convolving over images to harness the dimensional information available to extract features, before flattening the smaller, deeper matrix into an array, allows for the most efficient and effective identification of nuance and patterns by which objects can be classified.

Both networks were prone to high training times, MLP because of the exponential number of weights and calculations being made as each hidden layer was added, and CNN because of the myriad times that the (albeit) fewer parameters are used in training: A 3 channel input of shape ***(a, b, c)***, with ***i*** filters of shape ***(n, n, n)*** makes $P_{CNN} = 3i(n^3 + bias)$ parameters. To evaluate the layer, $a \cdot b \cdot c \cdot 3i(2n^3 + 1)$ operations need to be calculated, making an overall complexity of $O(abcP_{CNN})$. Use of Kingma and Ba's Adam optimization algorithm [10] also exacerbates computational workload. The algorithm uses moving averages of the parameters (e.g momentum) to calculate a scaled error gradient, to allow for a larger effective step size without needing to be tuned. Although observably beneficial in the present results, the repeated storing of the average and variance of each parameter effectively triples the size of the network being worked on [1][10].

Cifar 10 classification accuracy has neared 100% [6], and so future research could perhaps focus instead on the explainability of such models, improving the user's ability to see layer by layer how learning of features is taking place in order to then feedback this knowledge in to improving models and further broadening their potential uses as a consequence [14].

6. References

[1] Bengio, Y., 2012. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade* (pp. 437-478). Springer, Berlin, Heidelberg

[2] Browne, M.W., 2000. Cross-validation methods. *Journal of mathematical psychology*, *44*(1), pp.108-132.

[3] Çalik, R.C. and Demirci, M.F., 2018, October. Cifar-10 image classification with convolutional neural networks for embedded systems. In *2018 IEEE/ACS 15th*

[4] Chauhan, R., Ghanshala, K.K. and Joshi, R.C., 2018, December. Convolutional neural network (CNN) for image detection and recognition. In *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)* (pp. 278-282). IEEE.

[5] Gardner, M.W. and Dorling, S.R., 1998. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, *32*(14-15), pp.2627-2636.

[6] Han, K., Xiao, A., Wu, E., Guo, J., Xu, C. and Wang, Y., 2021. Transformer in Transformer. *arXiv preprint arXiv:2103.00112*.

[7] Heaton, J., 2008. *Introduction to neural networks with Java*. Heaton Research, Inc..

[8] Kanellopoulos, I. and Wilkinson, G.G., 1997. Strategies and best practice for neural network image classification. *International Journal of Remote Sensing*, *18*(4), pp.711-725.

[9] Krizhevsky, A. and Hinton, G., 2009. Learning multiple layers of features from tiny images.

[10] Kingma, D.P. and Ba, J., 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[11] Longadge, R. and Dongre, S., 2013. Class imbalance problem in data mining review. *arXiv preprint arXiv:1305.1707*.

[12] Lu, D. and Weng, Q., 2007. A survey of image classification methods and techniques for improving classification performance. *International journal of Remote sensing*, *28*(5), pp.823-870.

[13] Odense, S. and Garcez, A.D.A., 2020. Layerwise Knowledge Extraction from Deep Convolutional Networks. *arXiv preprint arXiv:2003.09000*.

[14] Zaniolo, L. and Marques, O., 2020. On the use of variable stride in convolutional neural networks. *Multimedia Tools and Applications*, *79*(19), pp.13581-13598.

[15] Zhang, C., Pan, X., Li, H., Gardiner, A., Sargent, I., Hare, J. and Atkinson, P.M., 2018. A hybrid MLP-CNN classifier for very fine resolution remotely sensed image classification. *ISPRS Journal of Photogrammetry and Remote Sensing*, *140*, pp.133-144.

Appendix

Larger images of two of the figures included below, in case they are hard to read.

As mentioned in the report, a limitation of this study was that some networks trained were not the result of a gridsearch algorithm, because of time constraints with regards to how long it took to train many different combinations of models. Hyperparameter ranges changed over grid-searches as well for the same reason. For clarity, the full list of hyperparameters trained in the grid-search stage are listed below. Hyperparameters that are not included in some grid-searches were made fixed values as a result of previous results indicating their best value and dropping them to focus on hyperparameters for which the optimum value was yet unknown. These hyperparameter values can also be seen in the training jupyter notebook. However, some values may appear to be not have been tested in the notebook, as a result of cleaning the notebook of initial exploratory methods of training, including some preliminary grid-searches, the results of which were recorded by hand. Deleted networks from this cleaning were then retrained by singular skorch classifier models with the best parameters input directly, if at all.

MLP:

1 hidden layer:

Hyperparameter	**Values**
Hidden layer 1 nodes	175, 250, 500, 1000
Dropout	0, 0.2, 0.5
Learning rate	0.0001, 0.001, 0.01, 0.1
Momentum	0.7, 0.9, 0.99

2 hidden layers:

Hyperparameter	**Values**
Hidden layer 1 nodes	250, 500, 1000
Hidden layer 2 nodes	500, 1000, 1500
Dropout	0, 0.2, 0.5
Learning rate	0.0001, 0.001, 0.01
Optimizer	Torch.optim.Adam, torch.optim.SGD

3 hidden layers:

Hyperparameter	**Values**
Hidden layer 1 nodes	2000, 1500, 1000
Hidden layer 2 nodes	750, 1000
Hidden layer 3 nodes	250, 500, 750
Dropout	0.2, 0.5
Learning rate	0.0001, 0.001
Momentum	0.7, 0.9

CNN:

2 conv layers:

Hyperparameter	**Values**
Conv layer 1 nodes	32, 64
Conv layer 2 nodes	64, 128
Fully connected layer 1	1500, 1000
Batch Size	32, 64

The larger CNNs were trained on a one-by-one basis, because grid-searches were proving far too computationally strenuous. More hyperparameters were experimented with in this way, including weight decay for the Adam optimizer, stride of middle hidden layers, different numbers of normalization and activation layers etc.

Glossary

Activation/non-linearity function – A function that is applied to data in a neural network in order to account for non-linear patterns/behaviour in machine learning. Using any activation function can assist a neural network in learning non-linear patterns, by acting on different values in different ways depending on their size.

Backpropagation – short for 'backward propagation of errors', this weight adjustment algorithm for supervised learning is present in both of the neural networks trained in the present paper, allowing their comparison. It works by computing the gradient of a loss function between the true value of an input's label and the predicted label produced from running the inputs through the weights of neural network. The weights are then adjusted accordingly in the direction of the gradient in the hope of reaching the global minima of error.

Convolutional window – Refers to the convolutional filter that is 'slid' across the image in a CNN.

Dropout – A normalization technique in which each node of a network is given a probability of being dropped out of the network along with its connections. This is done to prevent units from co-adapting too much/becoming too dependent on each other, and thus results in less overfitting to the training data.

Internal covariate shift - the change in the distribution of network inputs during the training of a neural network. This shift can be caused due to a non-representative train-test split, or due to the order in which data inputs are fed to a neural network in batches or mini-batches during training.

K-fold Cross validation – A method of validation of a model/network's ability to perform at whatever task it is attempting. A training set will be split in to k folds, and k models will be trained on k-1 folds of the data, before being tested on the k^{th} fold to give a score, be it classification accuracy, loss, or regression metrics etc. This is done k times, with all folds of the data being the validation set once. The scores from each run through are then averaged for a more robust measure of how well a model is expected to perform on unseen data.

Precision – The degree to which the positive classifications made are relevant/correct (true positives / all positives).

ReLU – A simple type of activation function: $y = max(0, x)$. This means that a value will be transformed to 0 if it is equal to or less than 0, and will remain as it is if it is above 0.

Sensitivity/recall – True positive rate – the ability of a test to correctly identify an instance of something, in the present case correctly classifying an object (true positives / (true positives + false negatives)).

Specificity – False positive rate – the ability of a test to correctly identify – in this case – that an object is *not* an instance of something e.g. correctly predicting that an object that is not a truck is indeed not a truck (true negatives / (true negatives + false positives)).

Weight – A number that is assigned to a connection between two nodes in a fully connected layer, or is often a product of several nodes in convolutional layers. It is simply a parameter within a network that transforms the input data as it is fed forward through the network to produce an output. The weights are adjusted in both of the above networks through backpropagation.