CIFAR-10 Classification With Multiple Deep Learning Models

Deep Learning

Master's degree in Artificial Intelligence [LM-18]

Daniele Pasotto

AY 2024/2025

Contents

1 Motivation and rationale

The project is an extension of the Machine Learning one, and in particular will cover the potentiality and the limitations of the various Deep Learning models present in the AI domain. It also focuses on the importance of applying data augmentation on a dataset to achieve better performances.

2 State Of The Art

The project contains the most common state-of-the-art deep learning approaches. In general, the various techniques adopted are CNN, RNN, Transformer and MLP architectures, Adam optimizer, regularization methods such as early stopping and preprocessing methods such as data augmentation.

3 Objectives

The main goal of this project is to analyze the power of data augmentation, so different DL models are trained with and without this technique to have a comparative view. Finally, the report will draw conclusions from the data obtained. In summary, the points that cover this work are:

- Train different DL models with only normalization;
- Train different DL models with data augmentation;
- Compare the performance on the models;
- Draw conclusions from the obtained data.

4 Methodology

4.1 Dataset

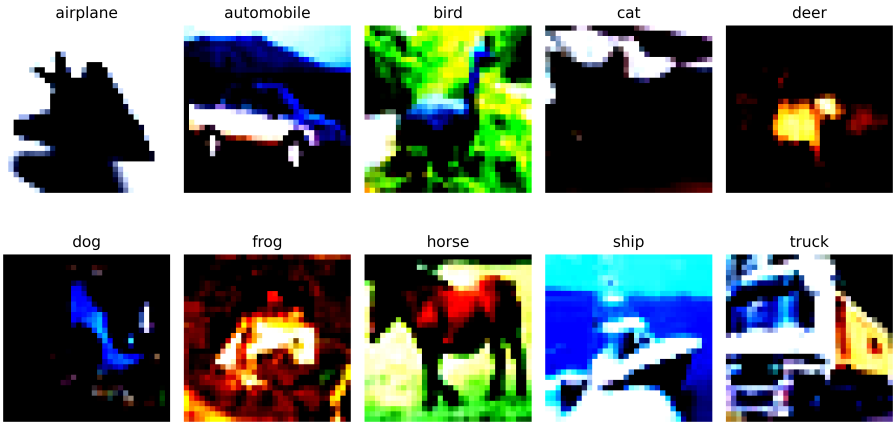
The dataset used is CIFAR-10[1], which consists of 60000 colour images of the size of 32x32. This dataset is often used as a benchmark dataset for various models, both in deep learning and machine learning. The dataset is composed of 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck), with 6000 images per class. In total, there are 50000 training images and 10000 test images. The classes are completely mutually exclusive (there is no overlap between classes that may be similar).

Figure 1: CIFAR-10 dataset

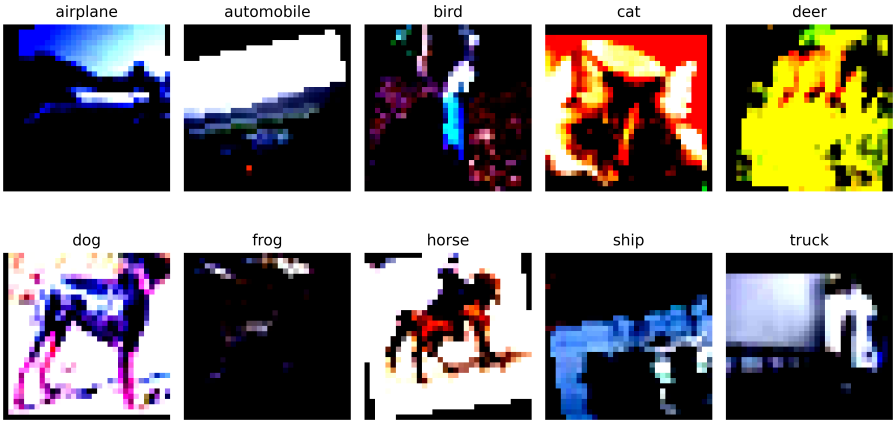
4.2 Preprocessing

The training and test sets are preprocessed by applying two type of techniques:

- The first one is simply a **normalization** of the images using the mean and standard deviation calculated from the initial training set.
- The second approach is the **data augmentation**, in particular on training images normalization, random crop, random horizontal flip, random rotation and color jitter are applied; whereas on test images only normalization is applied.



(a) Dataset with normalization



(b) Dataset with data augmentation

Figure 2: Dataset preproccessing

4.3 Training

The training process is composed of a total of 100 epochs and are involved the training and validation set. The validation set is calculated from a 20% split of the training set. The main purpose of this step is to find the best hyperparameters for each model, which maximize the accuracy of the predictions. Inside these hyperparameters there is also the choice of the best training batch size, which is very important in the increase of the model performances.

In the training phase an early stopping trick is implemented to avoid long executions when the convergence is reached very soon with respect to the maximum number of epochs set. In particular, this method stops the execution of the training if the loss doesn't improve after 5 epochs, with respect to the best loss find during the previous iterations. This trick is also useful to prevent overfitting during the training.

In order to find the best hyperparameters, the training process follows this pipeline:

1: Setting hyperparameter grid with model and batch size hyperparameters;
2: **for** combination of hyperparameter in hyperparameter grid **do**
3: Set the train loader with hyperparamter batch size;
4: Set the validation loader;
5: Set the model with its hyperparamters;
6: Set the criterion as Cross Entropy Loss;
7: Set the optimizer as Adam with 0.001 learning rate;
8: Train the model on the training loader and evaluate performance on the validation loader such as early stopping is triggered or the maximum number of epochs are reached;
9: **if** max(validation accuracy) > best accuracy **then**
10: best accuracy ← max(validation accuracy);
11: **end if**
12: Save the results of the training;
13: **end for**
14: Print the best hyperparameters which obtained the maximum accuracy on validation present in the results;

When the hyperparameters are founded, the model is trained using them and finally saved.

4.4 Models

The following models come from the most important families of deep learning models that already exist and are implemented from scratch.

Every model has its hyperparameters to be set, which are chosen using the methodology explained before in the training section. The selected hyperparameters are obtained from the training on the normalized training set and are used also for the training on the augmented training set.

4.4.1 TinyVGG

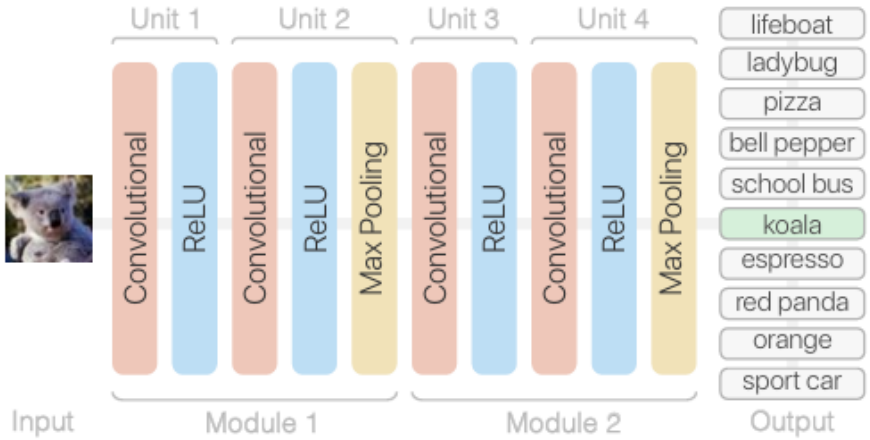


Figure 3: TinyVGG architecture

The TinyVGG[2] model is a lite version of the original VGGNet model and uses the same, but fewer, convolutional layers. It consists of two modules, each composed of two convolutional layers with 3x3 kernel and ReLU activations, followed by a max pooling layer.

Since the number of convolutional layers is fixed, the hyperparameters to tune are as follows:

- **Hidden units**: [32, 64, 128]
- **Batch size**: [32, 64, 128]

Applying the research of the best hyperparameters, the result is the following:

- **Hidden units**: 128
- **Batch size**: 128

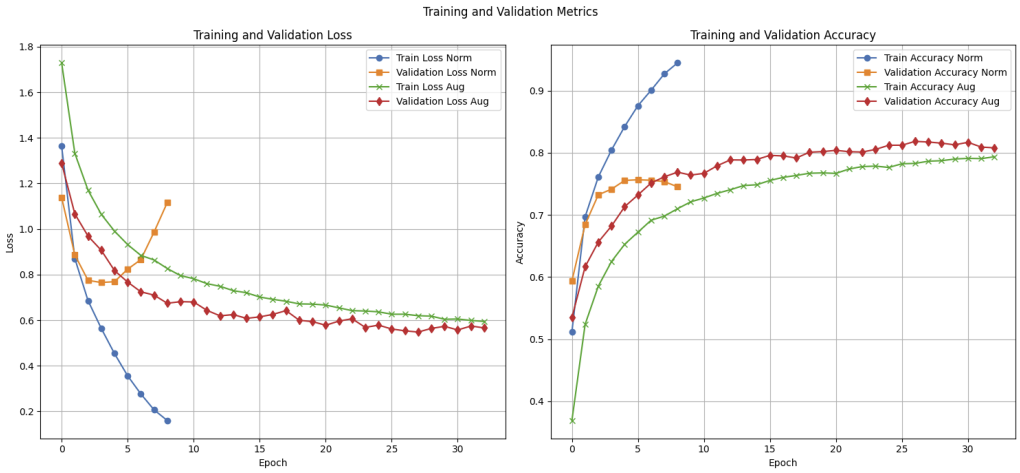


Figure 4: TinyVGG training result on normalized dataset and augmented dataset

Training Results For normalized training set the early stopping is triggered after 9 epochs, whereas for augmented one the early stopping is triggered after 33 epochs.

From the plots can be seen that for training with normalized dataset the risk of falling into overfitting is high and is contained with the help of early stopping. Instead with augmented dataset, the model generalizes better and improves performances on validation set.

4.4.2 ResNet-12

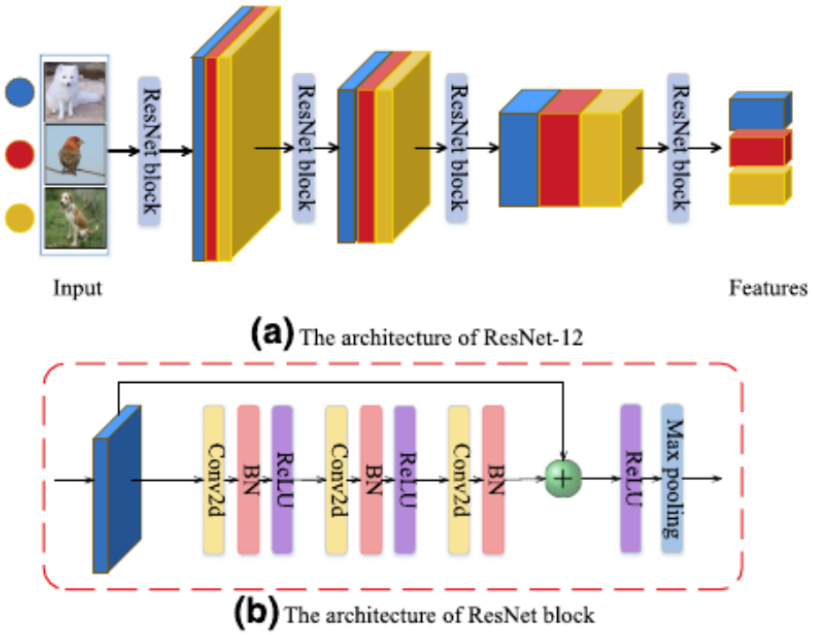


Figure 5: ResNet-12 architecture

The ResNet-12 model is a lightweight residual neural network architecture commonly used in few-shot learning and low-resource image classification tasks. It consists of

a total of 4 residual blocks, each composed of three convolutional layers 3x3 kernel, batch normalization and ReLU activations, skip connection and max pooling layer. Compared to deeper versions (like ResNet-50 or ResNet-101), ResNet-12 offers a good trade-off between depth and speed.

Since the number of convolutional layers and resnet blocks are fixed, the hyperparameters to tune are as follows:

- **Hidden units**: [32, 64, 128]
- **Batch size**: [32, 64, 128]

Applying the research of the best hyperparameters, the result is the following:

- **Hidden units**: 128
- **Batch size**: 64

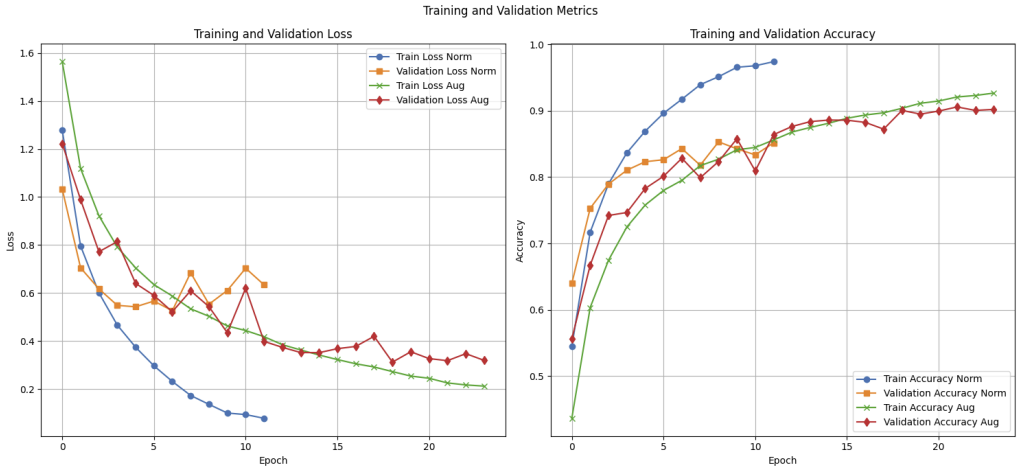


Figure 6: ResNet-12 training result on normalized dataset and augmented dataset

Training Results For normalized training set the early stopping is triggered after 12 epochs, whereas for augmented one the early stopping is triggered after 24 epochs.

The plots follow a pattern similar to TinyVGG results, but with a slight improvement of the performance.

4.4.3 GRU

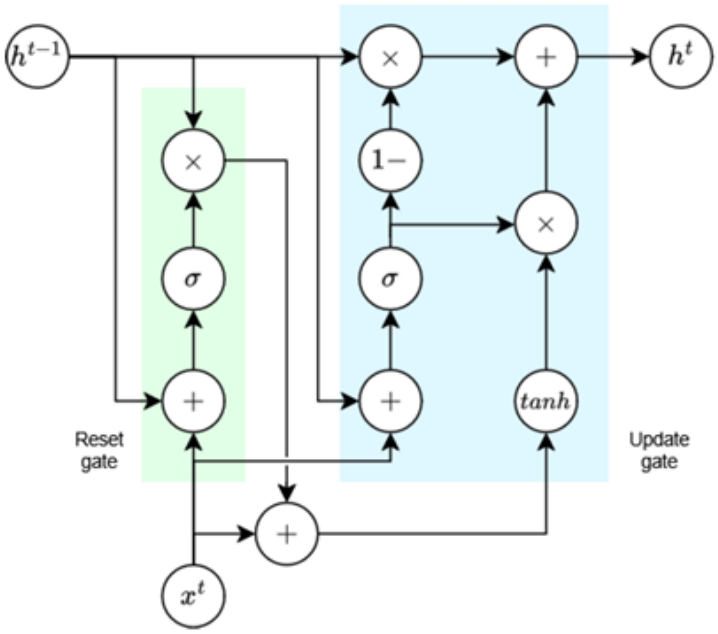


Figure 7: Gated Recurrent Unit architecture

A Gated Recurrent Unit[3], or GRU, is a type of recurrent neural network. It is similar to an LSTM, but only has two gates - a reset gate and an update gate - and notably lacks an output gate. Fewer parameters means GRUs are generally faster to train than their LSTM counterparts.

The hyperparameters to tune are as follows:

- **Hidden units**: [32, 64, 128]
- **Number of layers (how many GRUs stack together)**: [1, 2, 3, 4]
- **Batch size**: [32, 64, 128]

Applying the research of the best hyperparameters, the result is the following:

- **Hidden units**: 64
- **Number of layers**: 8
- **Batch size**: 32

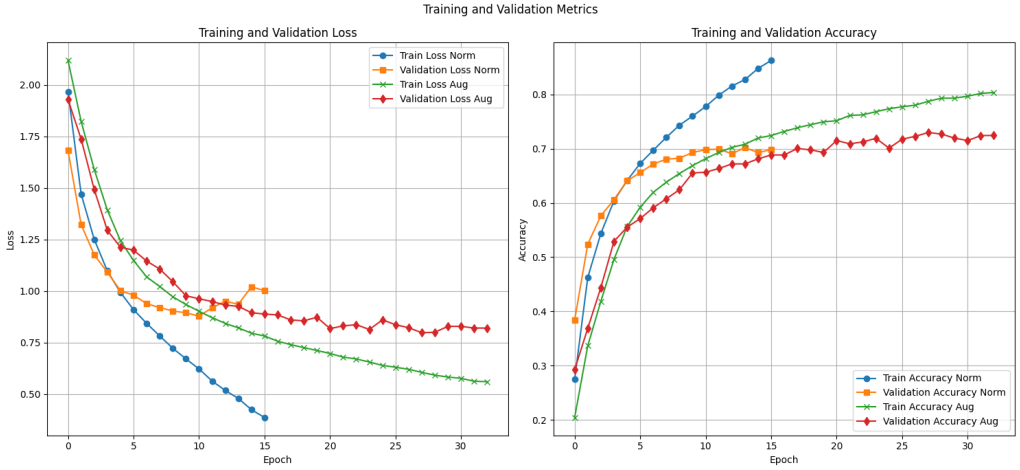


Figure 8: GRU training result on normalized dataset and augmented dataset

Training Results For normalized training set the early stopping is triggered after 16 epochs, whereas for augmented one the early stopping is triggered after 33 epochs.

The plots follow a pattern similar to the other model results, but can be seen smoother steps and worst performance in general.

4.4.4 Vision Transformer

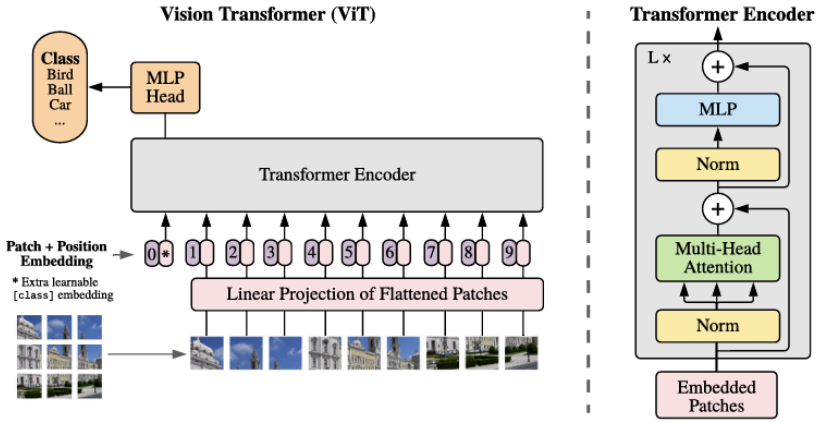


Figure 9: Vision Transformer architecture

The Vision Transformer[4] is a novel architecture that adapts the Transformer model, originally developed for natural language processing (NLP), to the domain of computer vision.

ViT essentially divides the input image into fixed-size patches, which are projected into a feature embedding. At this point a positional encoding is added to maintain spatial information, and the sequence is fed into a standard Transform encoder, which applies multi-head self-attention and feedforward layers. Finally a dense layer maps the output using the embedding of a special token.

Since the size of the patch (4), the number of heads for multi-head attention (8) and dropout (0.1) are fixed, the hyperparameters to tune are as follows:

- **Embed dimension**: [32, 64, 128]
- **Depth (number of Transformer encoder layers)**: [4, 6, 8]
- **MLP Dimension (number of hidden units for MLP)**: [128, 256, 512]
- **Batch size**: [32, 64, 128]

Applying the research of the best hyperparameters, the result is the following:

- **Embed dimension**: 64
- **Depth**: 8
- **MLP Dimension**: 512
- **Batch size**: 32

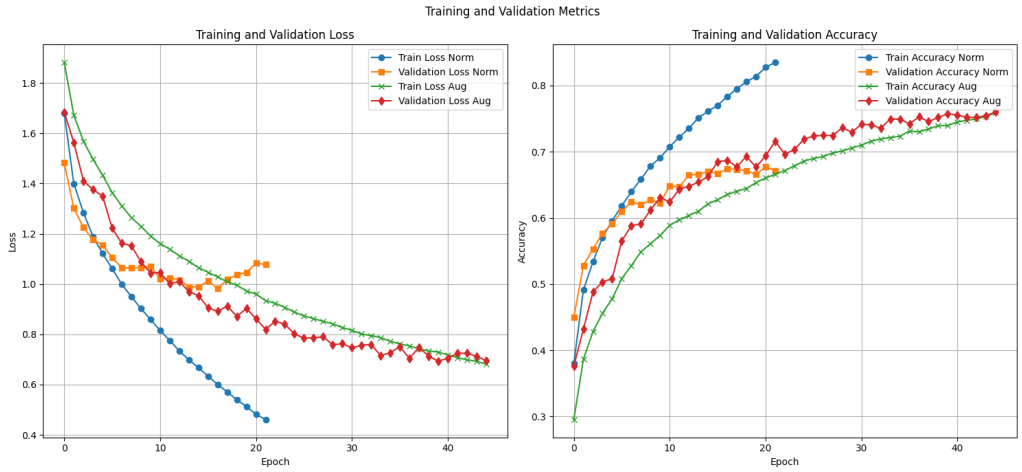


Figure 10: ViT training result on normalized dataset and augmented dataset

Training Results For normalized training set the early stopping is triggered after 22 epochs, whereas for augmented one the early stopping is triggered after 45 epochs.

The plots follow a pattern similar to the other model results. It's strange to see that the ViT model performs worse than other older architectures, because usually obtains superior performance in most of the cases, but the big limitation is that requires large training datasets and long training schedules. To improve the results, a solution is to use a pretrained ViT and fine-tune it on the CIFAR-10 dataset.

4.4.5 MLP-Mixer

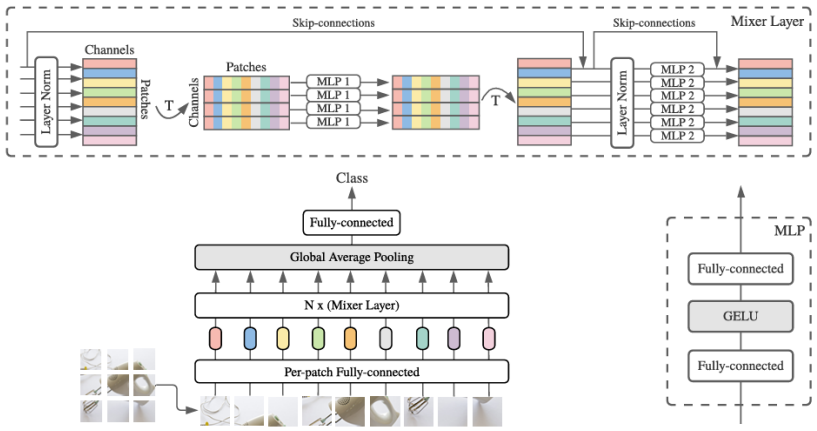


Figure 11: MLP-Mixer architecture

The MLP-Mixer[5] is a model introduced as non-convolutional and non-attention-based architecture for vision and sequence modeling tasks. The structure takes inspiration from ViT, but uses only multi-layer perceptrons(MLPs) for both spatial and feature mixing.

The MLP-Mixer divides the input into fixed-size patches, which are linearly projected into a feature embedding. These embeddings are stacked into a matrix. The model then passes the matrix to the Mixer Layer, composed of two blocks:

1. **Token-Mixing MLP**: Operates across patches (rows of the input matrix). Learns global interactions between positions (token/patches), treating each feature channel independently.

2. **Channel-Mixing MLP**: Operates across the feature dimensions (columns of the input matrix). Learns nonlinear feature combinations within each token.

Each block is preceded by layer normalization and includes residual connections and MLP composed of fully-connected layers and GELU[6] activation functions. Many Mixer Layer can be stacked, and the final output is averaged and classified using a standard fully-connected layer.

Since the size of the patch (4) and dropout (0.1) are fixed, the hyperparameters to tune are as follows:

- **Embed dimension**: [64, 128]
- **Depth**: [4, 6, 8]
- **Token intermediate dimension**: [64, 128]
- **Channel intermediate dimension**: [64, 128]
- **Batch size**: [32, 64, 128]

Applying the research of the best hyperparameters, the result is the following:

- **Embed dimension**: 64
- **Depth**: 6
- **Token intermediate dimension**: 64
- **Channel intermediate dimension**: 128
- **Batch size**: 32

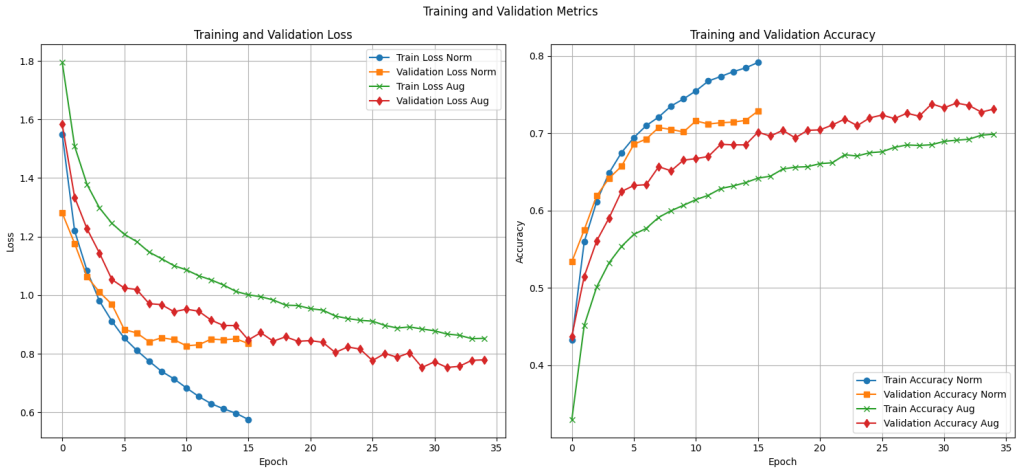


Figure 12: MLP-Mixer training result on normalized dataset and augmented dataset

Training Results For normalized training set the early stopping is triggered after 16 epochs, whereas for augmented one the early stopping is triggered after 35 epochs.

The plots follow a pattern similar to the other model results, but the training is visible more stable. An other thing that can be seen is that the validation performance with each preprocessing method is slightly the same, and it isn't strange because MLP-Mixer model is data-hungry and relies purely on token-mixing and channel-mixing MLPs without spatial locality or positional priors.

5 Analytical and Computation Tools

The major libraries used for the project are:

- `scikit-learn`: Used to calculate splittings and scores.
- `torch`: Used to implement all the deep learning models and techniques.
- `torchvision`: Used for dataset gathering and transform setting.
- `matplotlib`: Used to plot results in the form of graphs.

6 Results

6.1 Comparisons

Test Result with Normalization			
Model	**Loss**	**Accuracy**	**Time**
TinyVGG	1.081	75.82%	**3.438*s***
ResNet-12	**0.629**	**84.57%**	8.165*s*
GRU	1.033	68.44%	6.808*s*
ViT	1.105	67.06%	4.803*s*
MLP-Mixer	0.884	71.73%	3.646*s*

Table 1: Performance comparison of model trained with normalized data

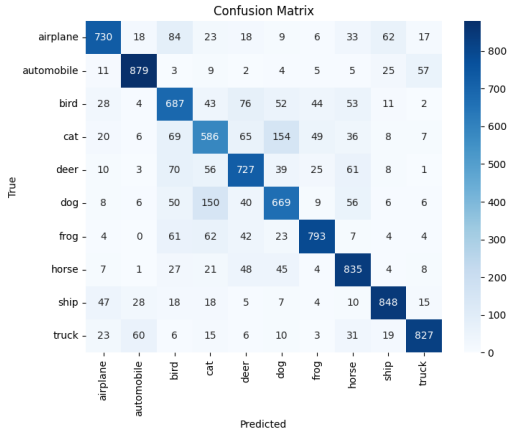
Test Result with Data Augmentation			
Model	**Loss**	**Accuracy**	**Time**
TinyVGG	0.568	81.09%	**3.251*s***
ResNet-12	**0.340**	**89.81%**	8.086*s*
GRU	0.845	71.47%	7.031*s*
ViT	0.712	75.09%	5.254*s*
MLP-Mixer	0.782	72.81%	3.673*s*

Table 2: Performance comparison of model trained with augmented data

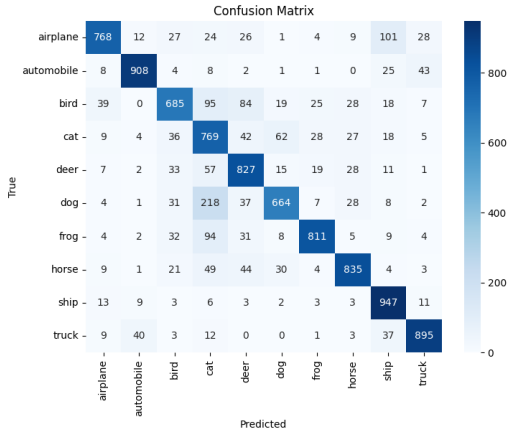
From the two tables can be seen that there is an overall improvement of the performance using data augmentation on dataset in the training phase. The unique exception is for the MLP-Mixer that has similar results for both setting, and the reason is mentioned in the section of its training result.

As expected, the two CNN models obtain very good performance, whereas the Transformer model is worse even if is a more recent and powerful architecture.

6.1.1 Confusion Matrix

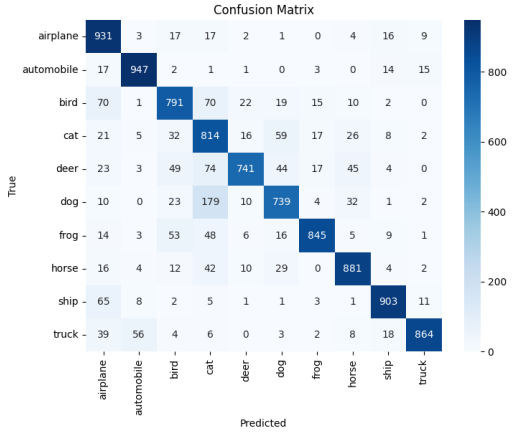


(a) TinyVGG trained on normalized dataset

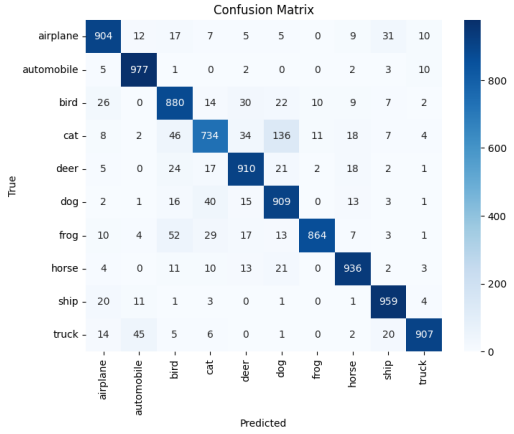


(b) TinyVGG trained on augmented dataset

Figure 13: TinyVGG confusion matrices

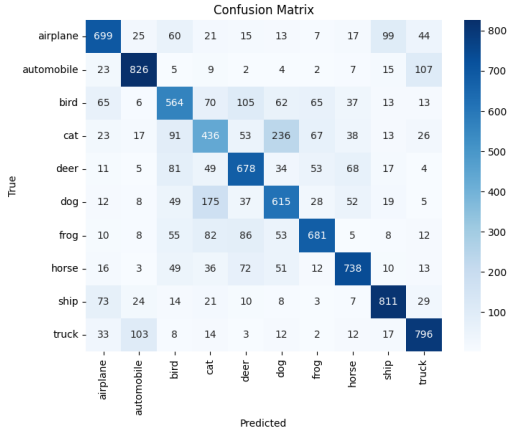


(a) ResNet-12 trained on normalized dataset

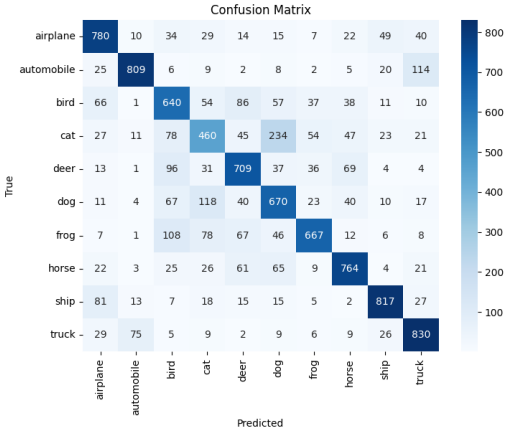


(b) ResNet-12 trained on augmented dataset

Figure 14: ResNet-12 confusion matrices

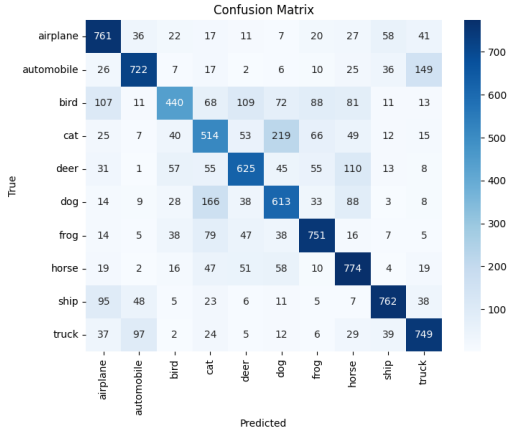


(a) GRU trained on normalized dataset

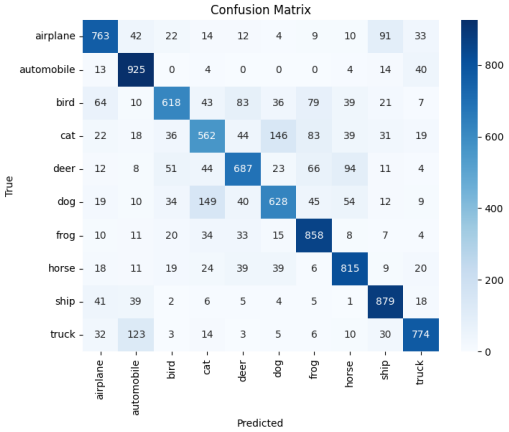


(b) GRU trained on augmented dataset

Figure 15: GRU confusion matrices

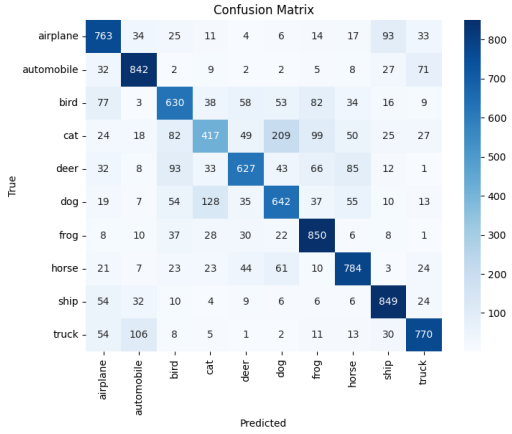


(a) ViT trained on normalized dataset

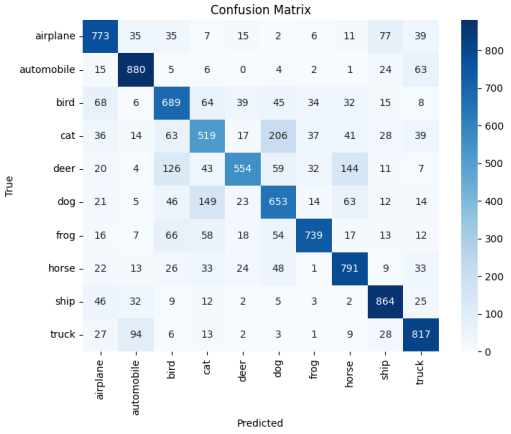


(b) ViT trained on augmented dataset

Figure 16: ViT confusion matrices



(a) MLP-Mixer trained on normalized dataset



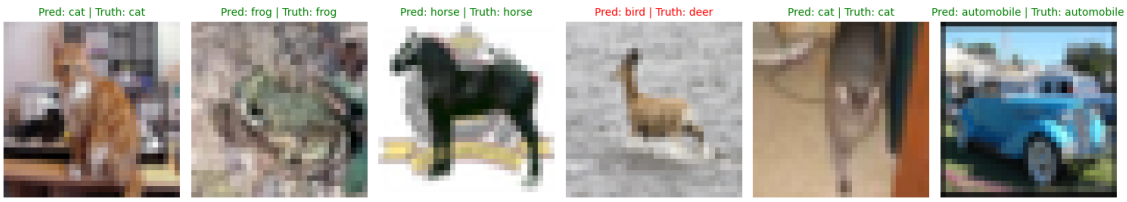
(b) MLP-Mixer trained on augmented dataset

Figure 17: MLP-Mixer confusion matrices

In general, from these confusion matrices can be seen that models fail to predict classes that are inside macro-classes (e.g. animal with another animal or vehicle with another vehicle). It is curious that in most of the cases it can be possible that there is a misclassification between bird and airplane. Most of the models fail to predict dog and cats correctly.

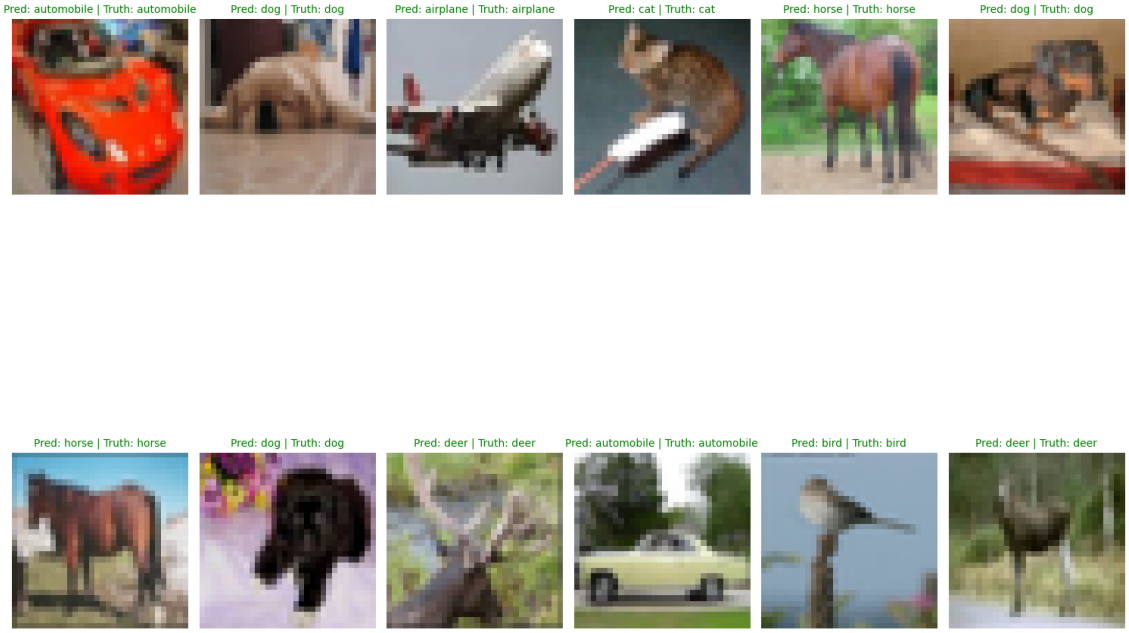
6.2 Predictions

(a) TinyVGG trained on normalized dataset

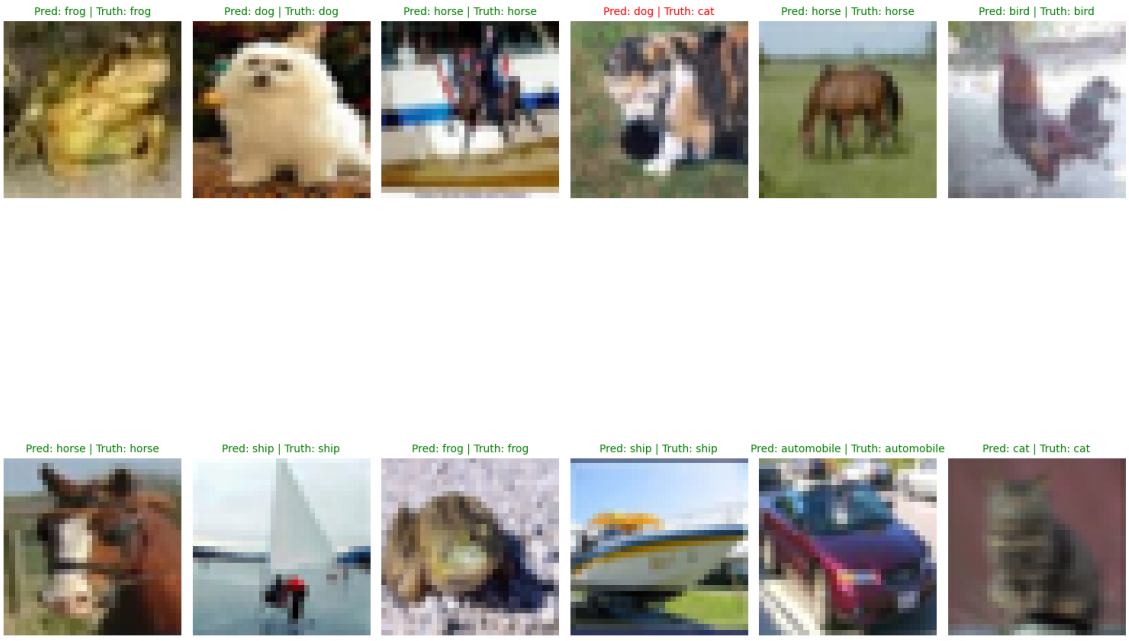


(b) TinyVGG trained on augmented dataset

Figure 18: TinyVGG predictions on test set

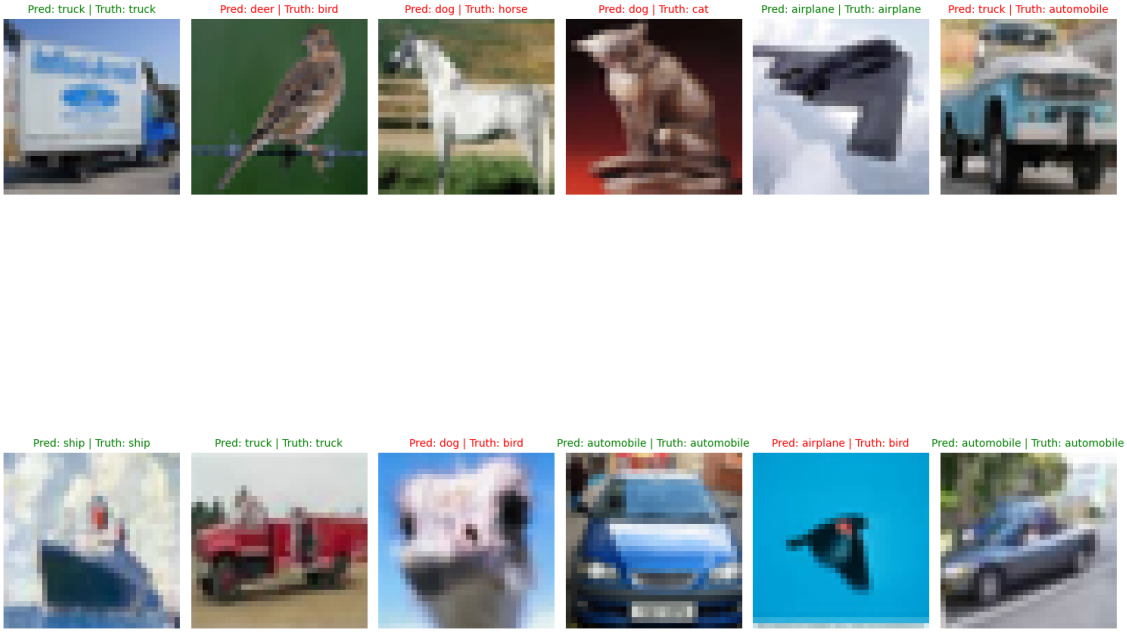


(a) ResNet-12 trained on normalized dataset

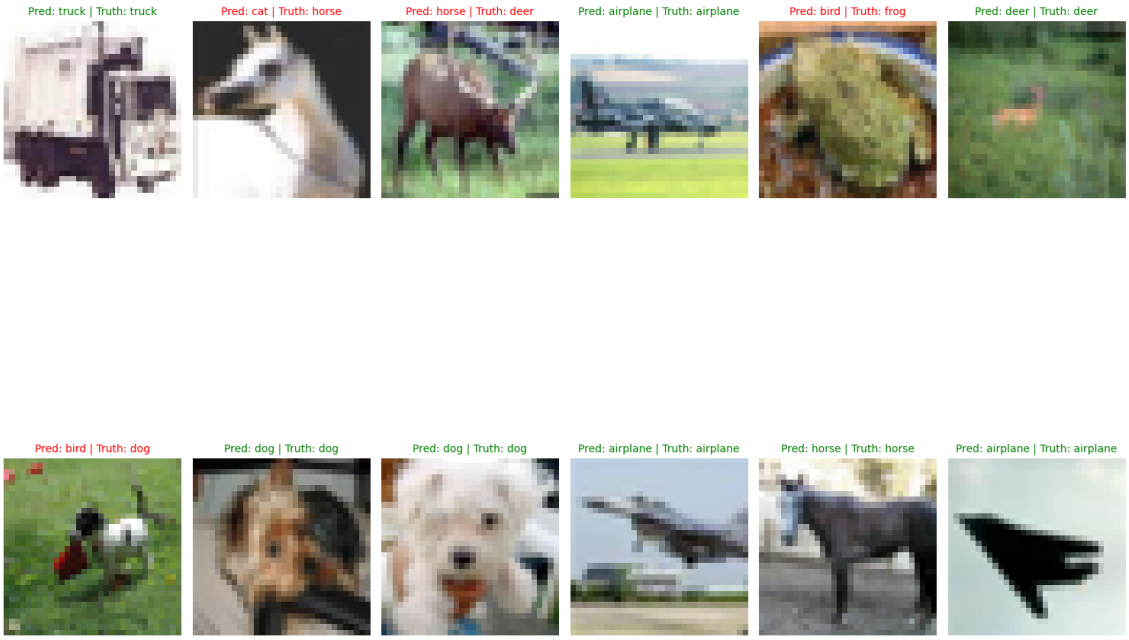


(b) ResNet-12 trained on augmented dataset

Figure 19: ResNet-12 predictions on test set

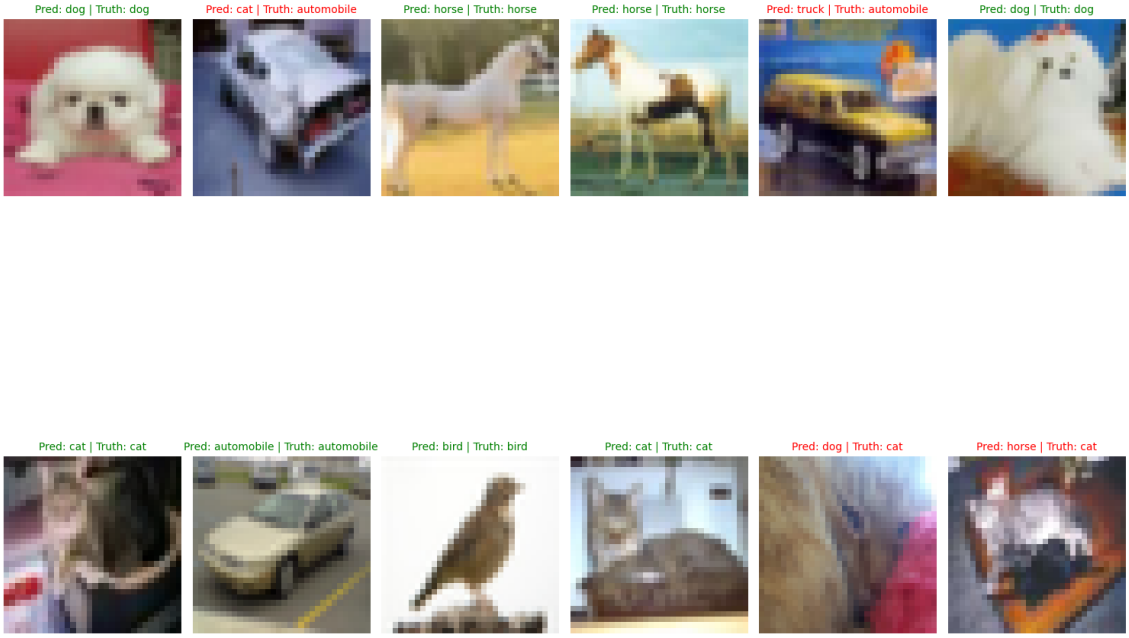


(a) GRU trained on normalized dataset

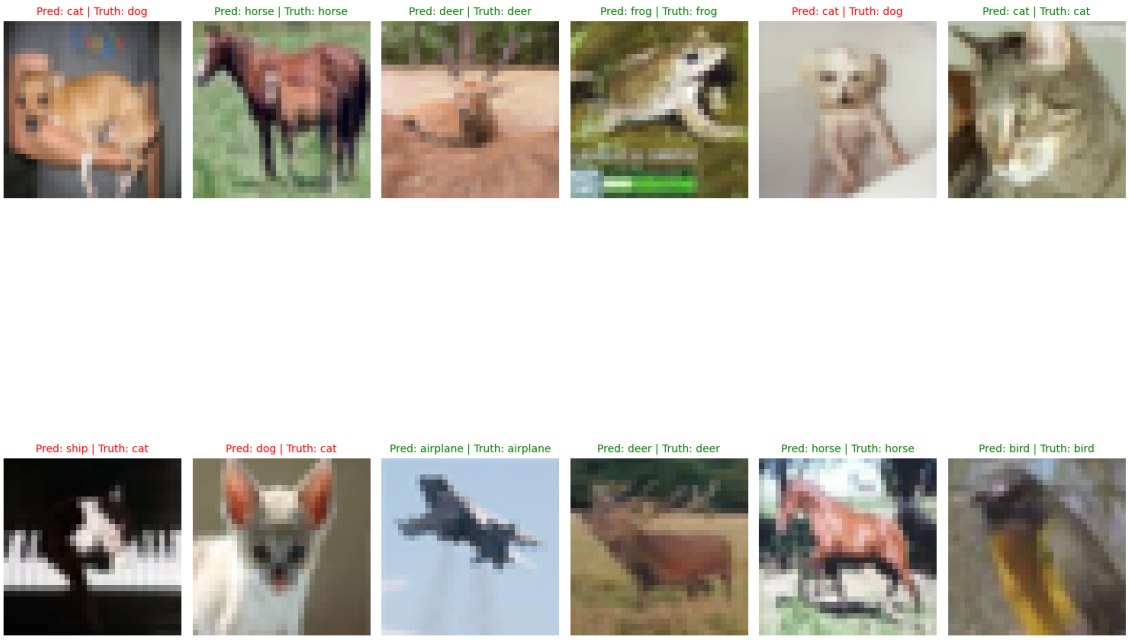


(b) GRU trained on augmented dataset

Figure 20: GRU predictions on test set

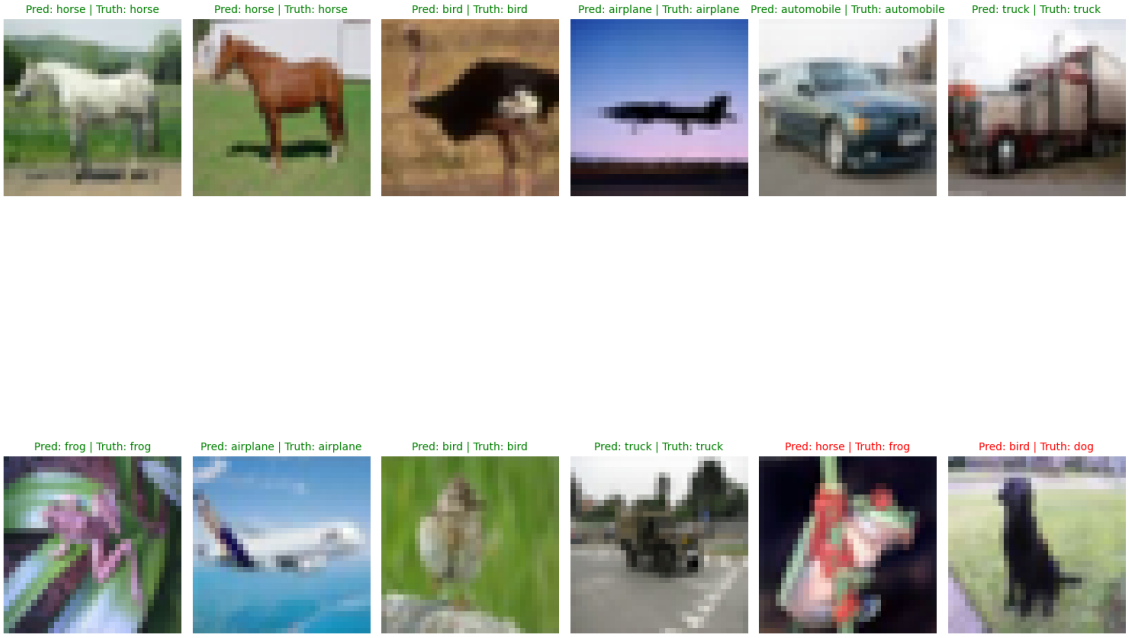


(a) ViT trained on normalized dataset

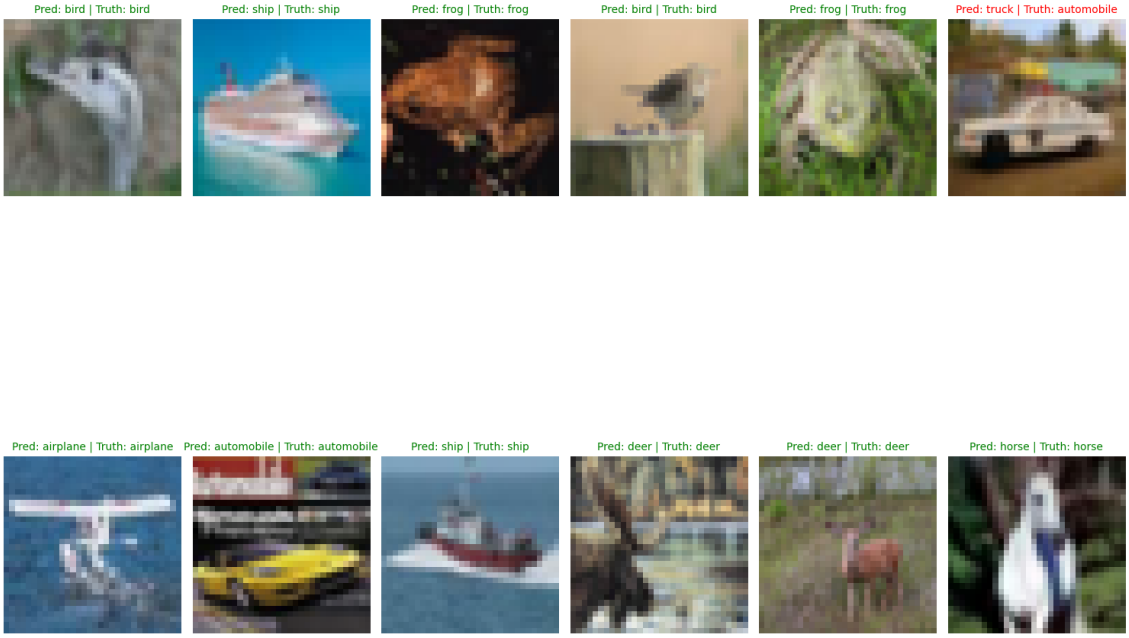


(b) ViT trained on augmented dataset

Figure 21: ViT predictions on test set



(a) MLP-Mixer trained on normalized dataset



(b) MLP-Mixer trained on augmented dataset

Figure 22: MLP-Mixer predictions on test set

7 Conclusions

Deep Learning models are very powerful and useful to perform feature extraction autonomously and to achieve very high performance in classification task. The disadvantage is that modern architectures need a huge amount of data to obtain good results, implying to have more powerful computational tools which are not so easy to obtain. The good trade-off remains for the most famous CNNs that also work with small datasets and achieve good performance.

References

[1] University of Toronto, *The CIFAR-10 dataset*. Available at this link: https://www.cs.toronto.edu/~kriz/cifar.html

[2] Z. J. Wang et al., "CNN Explainer: Learning Convolutional Neural Networks with Interactive Visualization", in IEEE Transactions on Visualization and Computer Graphics, vol. 27, no. 2, pp. 1396-1406, Feb. 2021, doi: 10.1109/TVCG.2020.3030418. https://arxiv.org/pdf/2004.15004

[3] J. Chung et al., "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling", Dec. 2014. https://arxiv.org/pdf/1412.3555

[4] A. Dosovitskiy et al., "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", Jun. 2021. https://arxiv.org/pdf/2010.11929

[5] I. Tolstikhin et al., "MLP-Mixer: An all-MLP Architecture for Vision", Jun. 2021. https://arxiv.org/pdf/2105.01601

[6] Dan Hendrycks, Kevin Gimpel, "Gaussian Error Linear Units (GELUs)", Jun. 2016. https://arxiv.org/pdf/1606.08415