University of Moratuwa
Faculty of Engineering
Department of Electronic & Telecommunication Engineering
Semester 5 (Intake 2020)

EN3150 - Pattern Recognition

Assignment 03: Simple convolutional neural network to perform classification.

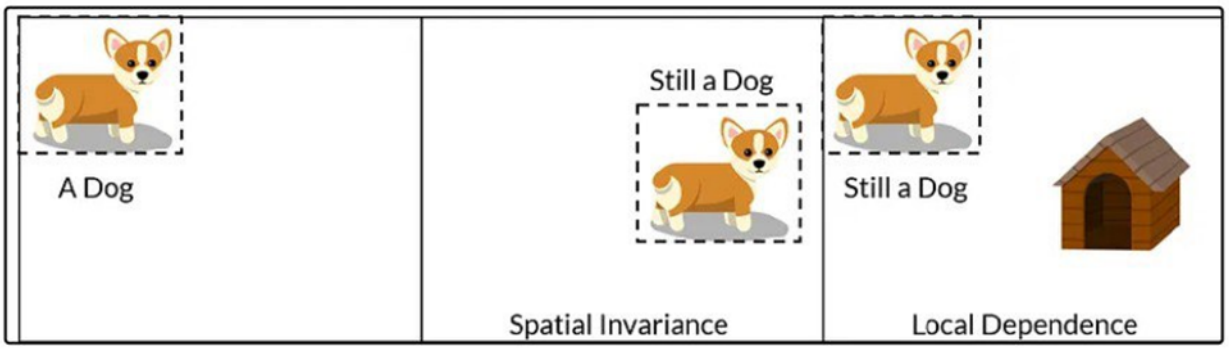
Group Members(**Team NeuroCraft**)	
Index No.	Name
200358G	LUCKSHAN G.W.C.M.
200397A	MIRIHAGALLA M.K.D.M.
200476P	PRAMUDITHA A.A.H.
200529H	RATHNASEKARA T.S.

This document is submitted as a partial fulfilment of module EN3150 - Pattern Recognition
November 5, 2023

CNN for image classification

Q. Why CNNs preferable for image classification over multilayered perceptrons (MLPs) or simple feedforward neural networks (NNs)?

Spatial Hierarchy: CNNs were originally designed to recognize spatial patterns of an image. They use convolutional layers (kernels) to recognize local patterns and hierarchically combine them to identify higher-level features. But MLPs identifies input data as flat vectors which will cause to lose gradient information of an image and poor performance in identifying edges and corners precisely.
Translation Invariance: CNNs can recognize image features regardless of their locations by using shared filters in convolutional layers. MLPs may struggle in recognizing the same feature in different locations of the image.
Pooling: CNNs use pooling layers to reduce spatial dimensions and maintain translation layers which MLPs don't have. So, they require more layers and parameters for similar results.



Q. Determine the parameters of the above network such as kernel sizes, filter sizes, size of the fully connected layer and dropout rate.

```
Model: "sequential_28"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_1974 (Conv2D)        (None, 32, 32, 128)       3584

 max_pooling2d_129 (MaxPooli  (None, 16, 16, 128)      0
 ng2D)

 conv2d_1975 (Conv2D)        (None, 16, 16, 64)        73792

 max_pooling2d_130 (MaxPooli  (None, 8, 8, 64)         0
 ng2D)

 dropout_9 (Dropout)         (None, 8, 8, 64)          0

 conv2d_1976 (Conv2D)        (None, 8, 8, 32)          18464

 max_pooling2d_131 (MaxPooli  (None, 4, 4, 32)         0
 ng2D)

 flatten (Flatten)           (None, 512)               0

 dense_38 (Dense)            (None, 512)               262656

 dropout_10 (Dropout)        (None, 512)               0

 dense_39 (Dense)            (None, 10)                5130

=================================================================
Total params: 363,626
Trainable params: 363,626
Non-trainable params: 0
_________________________________________________________________
```

Kernel size $= 3 \times 3$
Filter sizes used $= 32,\ 64,\ 128$
Sizes of the fully connected (dense) layers: 512 with ReLu, 10 with SoftMax

Q. Train the model: Train the model using the training data for 20 epochs and plot training and validation loss for with respect to epoch.

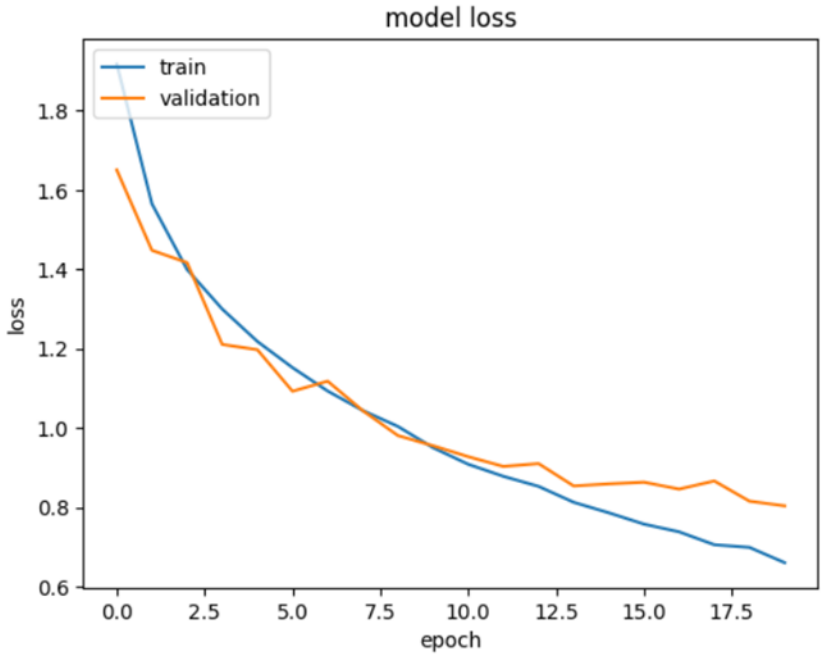


Figure 1: *Learning rate = 0.001*

Q. Why we have chosen adam optimizer over SGD?

Adam optimizer uses adaptive learning rates based on historical gradient information, enabling faster convergence and better solution in cases with varying gradient magnitudes. In SGD, initially, we must specify the learning rate, which won't be updated adaptively.
Adam optimizer uses gradient scaling feature in backpropagation to stabilize training process which will eventually reduce the manual tuning of the learning rate in SGD algorithm.

Q. Why we have chosen sparse categorical crossentropy as the loss function?

$$Loss = -\sum_{i=1}^{output\ size} y_i \cdot \log \hat{y}_i$$

Sparse Categorical Cross Entropy is a commonly used loss function in multi-class classification problems. This uses integer encoding which is more efficient when we have a lot of categories or labels which would consume a huge amount of resources (RAM).

Q. Evaluate the Model: After training, evaluate the model's performance on the testingdataset. Record the train/test accuracy, confusion matrix, precision and recall.

Test Accuracy of the custom CNN model:
F1 score: 0.720739974640534
Precision score: 0.7221389813500078
Recall score: 0.7208060245560287

Epochs	Training accuracy	Validation accuracy
1	0.282028	0.404917
2	0.432889	0.492333
3	0.493778	0.526500
4	0.528500	0.553250
5	0.567056	0.597917
6	0.591056	0.619833
7	0.609972	0.600333
8	0.622806	0.624583
9	0.638250	0.641250
10	0.654361	0.669083
11	0.668556	0.663083
12	0.685139	0.688417
13	0.695472	0.694750
14	0.708889	0.702833
15	0.717944	0.698083
16	0.727639	0.712500
17	0.737944	0.704750
18	0.736417	0.723167
19	0.750278	0.719500
20	0.757722	0.724500

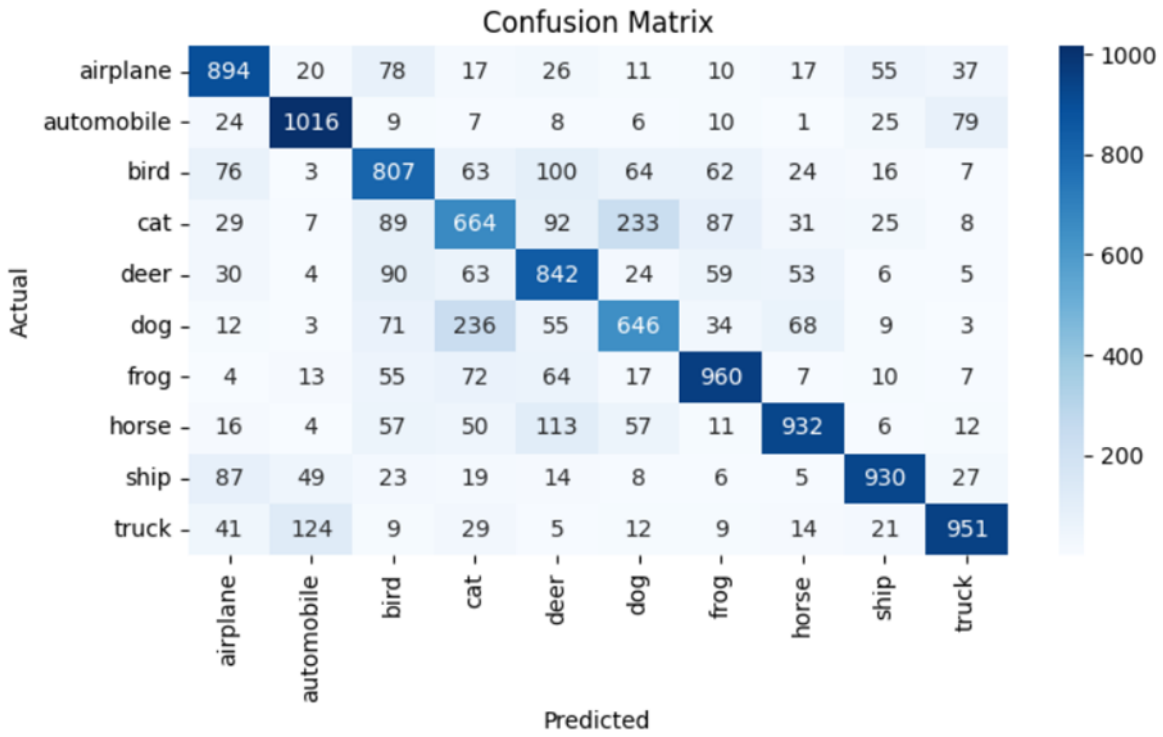
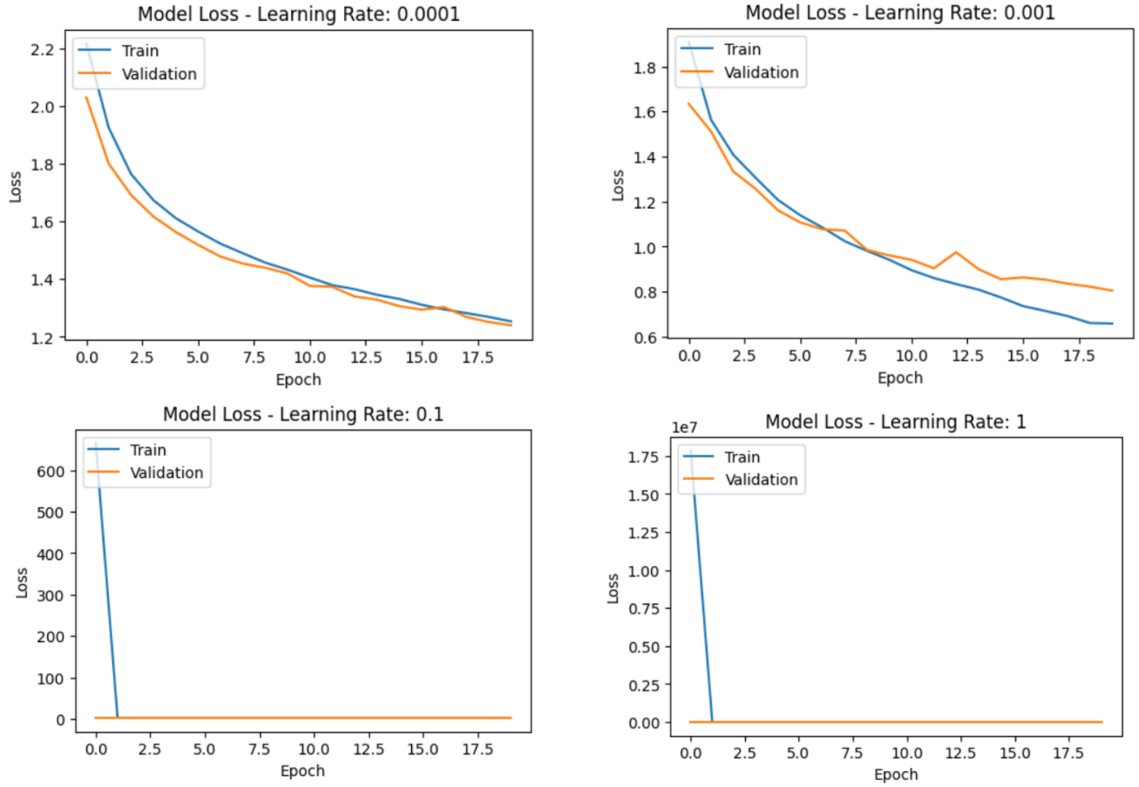


Figure 2: Confusion matrix

Q. Plot training and validation loss for with respect to epoch for different learning rates such as 0.0001, 0.001, 0.01, and 0.1



Compare your network with state-of-the-art networks

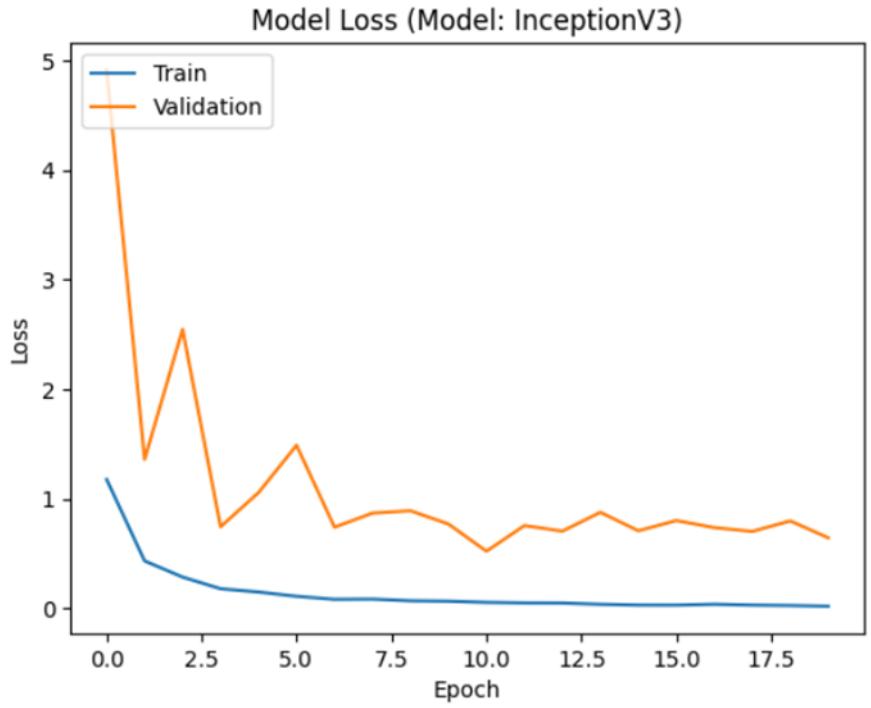
Q. Choose two state-of-the-art pre-trained model or architecture like ImageNet, ResNet, Googlenet, AlexNet, DenseNet and VGG

Used state-of-the-art model: GoogleNet (InceptionV3), ResNet

Q. Record training and validation loss values for each epoch.

Epochs	Training loss	Validation loss
1	1.179077	4.913738
2	0.435957	1.361068
3	0.288426	2.548080
4	0.182221	0.747121
5	0.152071	1.059078
6	0.113213	1.490367
7	0.085699	0.743338
8	0.087716	0.871771
9	0.072466	0.893217
10	0.068477	0.772433
11	0.058036	0.523935
12	0.052963	0.758343
13	0.052126	0.707310
14	0.040380	0.879242
15	0.033269	0.710570
16	0.032841	0.804128
17	0.040664	0.739373
18	0.033150	0.704940
19	0.029264	0.800657
20	0.022745	0.646480

Q. Evaluate the fine-tuned model on the testing dataset and calculate the test accuracy



Evaluations:
F1 score: 0.8638920339580709
Precision score: 0.8691847012328655
Recall score: 0.8643686317636835

Q. Compare the test accuracy of your custom CNN model with that of the fine-tuned state-of-the-art model.

Test Accuracy of the custom CNN model:
F1 score: 0.720739974640534
Precision score: 0.7221389813500078
Recall score: 0.7208060245560287

According to the above-mentioned statistics, it can be clearly seen that our fine-tuned state-of-the-art model gives higher test accuracy values when compared with the values given by the custom CNN model. Unlike our custom CNN model, the state-of-the-art model processes the dataset thoroughly and classifies them into the respective categories more precisely. Therefore, **State-of-the-art models** are often preferred, but custom models can be beneficial for specialized or domain-specific applications.

Q. Discuss trade-offs, advantages, and limitations of using a custom model versus a pre-trained model.

Custom Model	**Pre-Trained Model**
We can implement custom changes to the model as we want. We have the complete control over the architecture, hypermeters, and training process allowing us to optimize the model for your exact requirements.	Here, we are given a predefined architecture, and we have less control over the implementation and hyperparameter tuning operations.
Ability to address tiny differences and wants for our use case.	They have learned useful features from a wide range of data, which often leads to good performance when we have limited data for a specific problem.
Custom model training often requires a wide and varied dataset, which can be challenging and costly to gather and preprocess.	Pre-trained models can be readily available and easy to integrate into our applications.

*********END**********