PRDL -
Principles of Deep Learning

PROJECT PRESENTATION

06-DEC-2018

Presented By:

Lim Yuan Her (1881120H)

Agenda

- **Introduction**
  - ❑ Background
- **Data Preparation**
  - ❑ Data Preprocessing
- **Model**
  - ❑ Model Building
  - ❑ Model Training
  - ❑ Model Tuning
- **Conclusion**
- **References**

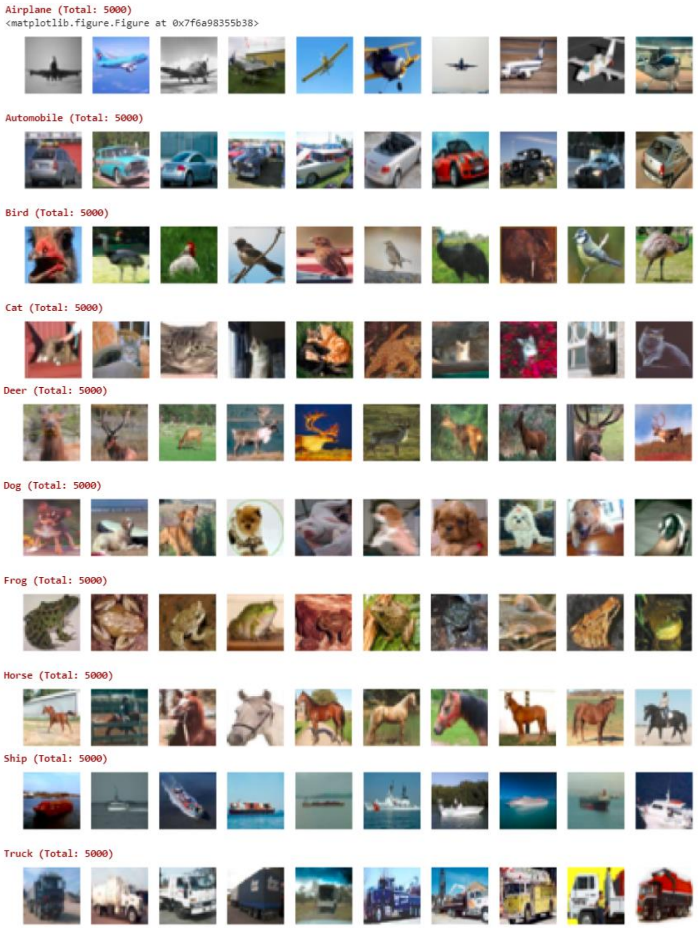
Introduction

- CIFAR-10 dataset from
https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz

- Consists of 60000 32x32 colour images
  - 50000 training and 10000 test images
  - 6000 images per class

- Total 10 Classes

Class Names	
Airplane	Dog
Automobile	Frog
Bird	Horse
Cat	Ship
Deer	Truck



Data Preparation

- Data Preprocessing

  1. Data Value Normalization
     - normalize the pixel values (0 to 255)
     - Involves transforming the original image in range of 0 to 1 (inclusive)

  2. One-Hot Encoding for categorical labels
     - Convert class vectors to binary class matrices

index	label
0	**airplane** (0)
1	**automobile** (1)
2	**bird** (2)
3	**cat** (3)
4	**deer** (4)
5	**dog** (5)
6	**frog** (6)
7	**horse** (7)
8	**ship** (8)
9	**truck** (9)
...	...
...	...

original label data

label	index											
	0	1	2	3	4	5	6	7	8	9	...	...
airplane	**1**	0	0	0	0	0	0	0	0	0	...	...
automobile	0	**1**	0	0	0	0	0	0	0	0	...	...
bird	0	0	**1**	0	0	0	0	0	0	0	...	...
cat	0	0	0	**1**	0	0	0	0	0	0	...	...
deer	0	0	0	0	**1**	0	0	0	0	0	...	...
dog	0	0	0	0	0	**1**	0	0	0	0	...	...
frog	0	0	0	0	0	0	**1**	0	0	0	...	...
horse	0	0	0	0	0	0	0	**1**	0	0	...	...
ship	0	0	0	0	0	0	0	0	**1**	0	...	...
truck	0	0	0	0	0	0	0	0	0	**1**	...	...

one-hot-encoded label data

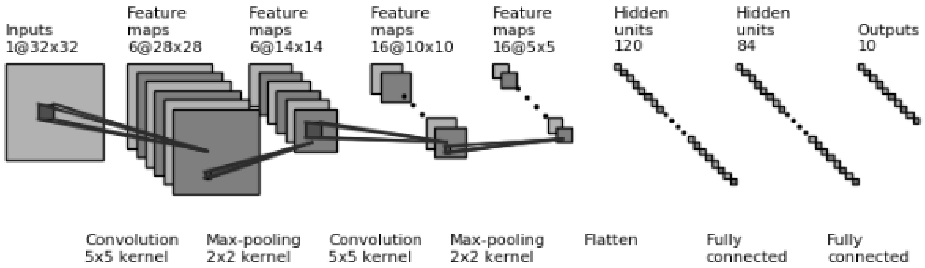
(Source: https://towardsdatascience.com/cifar-10-image-classification-in-tensorflow-5b501f7dc77c)

Model Building

- ❑ Build **<u>LeNet</u>** convolution neural network based on paper titled “Gradient-Based Learning Applied to Document Recognition” in 1998 by Yann Lecun, Leon Bottou, Yoshua Bengio and Patrick Haffner

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 32, 32, 6)	456
activation_5 (Activation)	(None, 32, 32, 6)	0
max_pooling2d_3 (MaxPooling2	(None, 16, 16, 6)	0
conv2d_5 (Conv2D)	(None, 12, 12, 16)	2416
activation_6 (Activation)	(None, 12, 12, 16)	0
max_pooling2d_4 (MaxPooling2	(None, 6, 6, 16)	0
conv2d_6 (Conv2D)	(None, 2, 2, 120)	48120
activation_7 (Activation)	(None, 2, 2, 120)	0
flatten_2 (Flatten)	(None, 480)	0
dense_3 (Dense)	(None, 84)	40404
dense_4 (Dense)	(None, 10)	850
activation_8 (Activation)	(None, 10)	0

Total params: 92,246
Trainable params: 92,246
Non-trainable params: 0

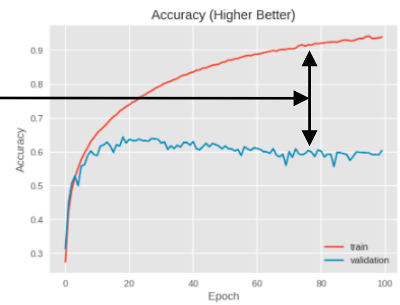


- ❑ Evaluation criteria:
  - Accuracy (higher is better)
  - Loss (lower is better)

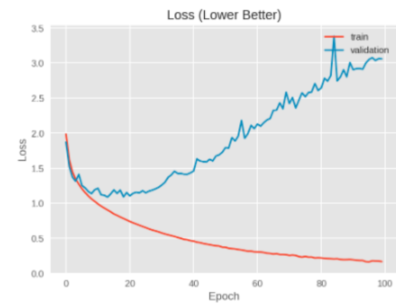
Model Training - 1

- Base LeNet model
  - Observed training overfitting at higher epochs

Significant gap between training and test accuracy profiles (obvious overfitting in trained model)



Accuracy

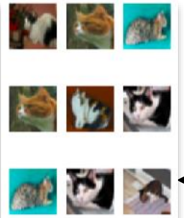


Loss

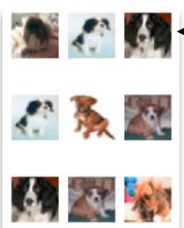
```
Test loss: 3.059764324569702
Test accuracy: 0.6032
[3.059764324569702, 0.6032]
```

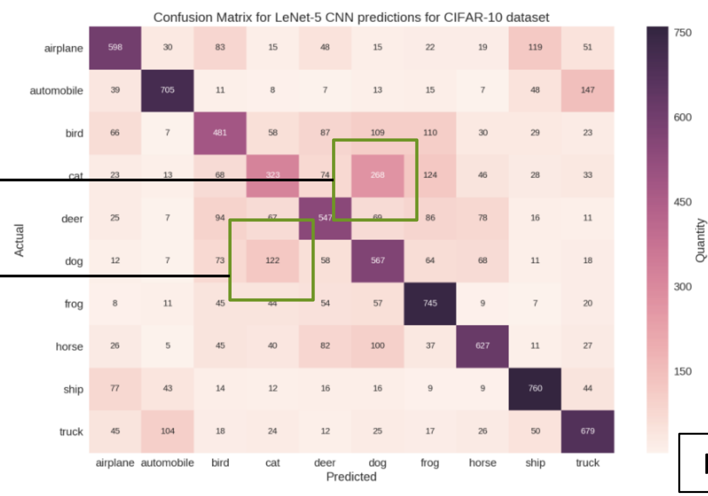
  - Observe higher incidences of mis-classification between Cat/Dog from the Confusion Matrix and classification report

Cats misclassified as dogs



Dogs misclassified as cats





	precision	recall	f1-score	support
airplane	0.65	0.60	0.62	1000
automobile	0.76	0.70	0.73	1000
bird	0.52	0.48	0.50	1000
cat	0.45	0.32	0.38	1000
deer	0.56	0.55	0.55	1000
dog	0.46	0.57	0.51	1000
frog	0.61	0.74	0.67	1000
horse	0.68	0.63	0.65	1000
ship	0.70	0.76	0.73	1000
truck	0.64	0.68	0.66	1000
avg / total	0.60	0.60	0.60	10000

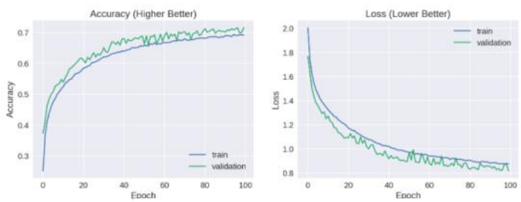
Performs best at classification of **automobiles** and **ships**

Model Training - 2

- Evaluation using various methods for overcoming overfitting problem
  - Dropout, L1/L2 regularization, Data Augmentation, Early Stopping, Mixed

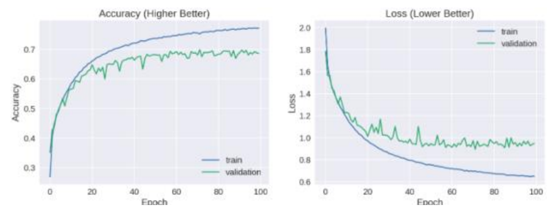
Shows significant improvements in accuracy/loss profiles

Dropout



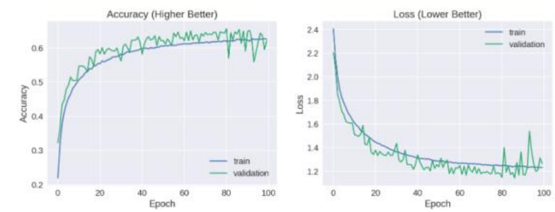

```
Test loss: 0.8184035935401917
Test accuracy: 0.7161
[0.8184035935401917, 0.7161]
```

Data Augmentation



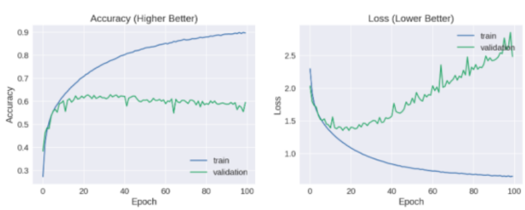

```
Test loss: 0.8863858350753784
Test accuracy: 0.7107
[0.8863858350753784, 0.7107]
```

Mixed



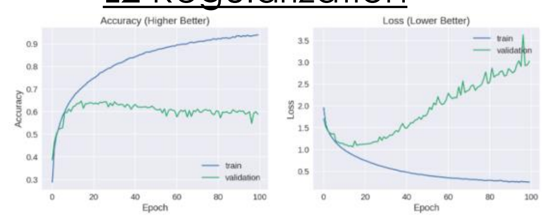

```
Test loss: 1.2028385154724122
Test accuracy: 0.6471
[1.2028385154724122, 0.6471]
```

L1 Regularization



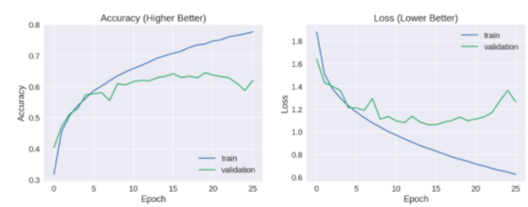

```
Test loss: 2.4853762031555178
Test accuracy: 0.5954
[2.4853762031555178, 0.5954]
```

L2 Regularization




```
Test loss: 3.0266742586135864
Test accuracy: 0.5888
[3.0266742586135864, 0.5888]
```

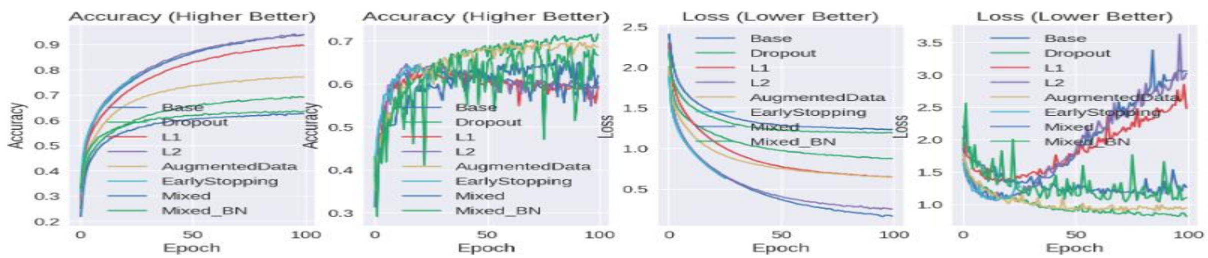
Early Stopping




```
Test loss: 1.26560675201416
Test accuracy: 0.6205
[1.26560675201416, 0.6205]
```

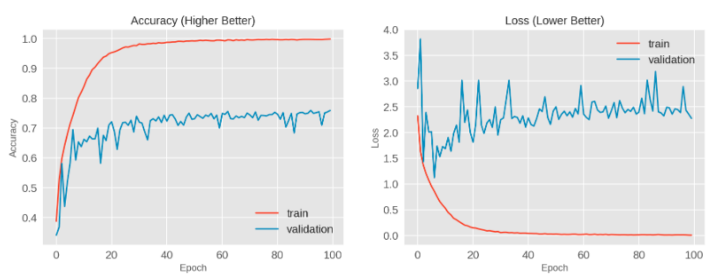
Model Training - 2

- Base/ Regularization Model Comparison
  - **<u>Dropout</u>** achieves the best accuracy/ loss profile



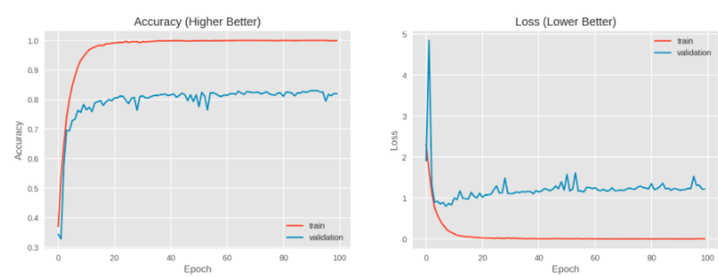
- Other Models (AlexNet/ VGG16)
  - Improved accuracies as compared to using LeNet model

<u>AlexNet</u>




```
Test loss: 2.2762546327590942
Test accuracy: 0.7586
[2.2762546327590942, 0.7586]
```

<u>VGG-16</u>




```
Test loss: 1.2190422409057617
Test accuracy: 0.8195
[1.2190422409057617, 0.8195]
```

Model Tuning – 1

- Use hyper-parameter tuning to improve the performance of the **LeNet CNN**

- Parameters Tuned:

  - **Batch Size** (batch_size) - number of patterns shown to the network before the weights are updated

  - **Training Optimization Algorithm** (optimizer) - different weight values

  - **Network Weight Initialization** (kernel_initializer)

  - **Neuron Activation Function** (activation) - controls the non-linearity of individual neurons and when to fire

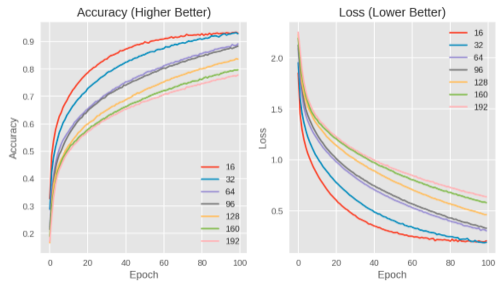
  - **Number of Neurons in the Hidden Layer** (neurons) - controls the representational capacity of the network

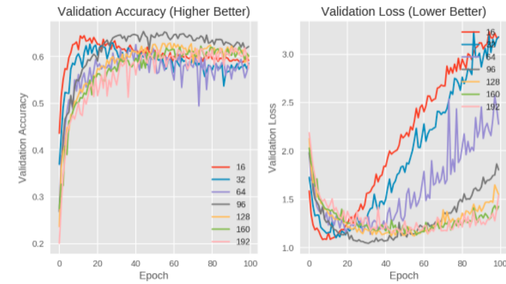
Model Tuning – 2

- Manual tuning
  - Select optimal parameter based on accuracy/ loss profiles

1. Batch Size
   - Optimal: **16**

Training Accuracy (Left)/ Loss (Right) Profiles

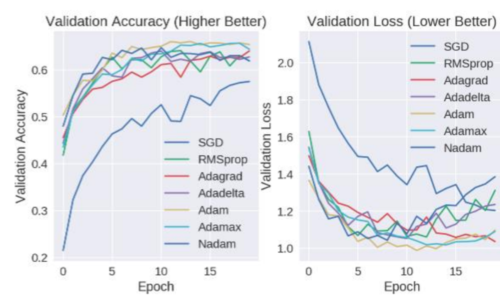




Testing Accuracy (Left)/ Loss (Right) Profiles

2. Training Optimization Algorithm
   - Optimal: **Adagrad**

Training Accuracy (Left)/ Loss (Right) Profiles

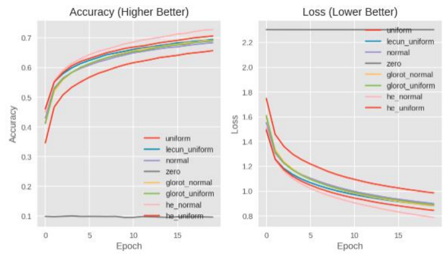


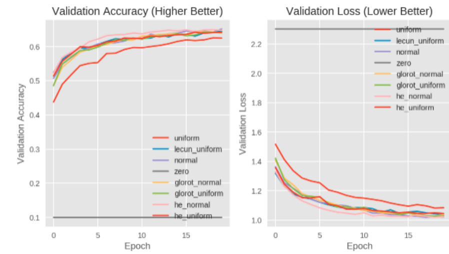
Testing Accuracy (Left)/ Loss (Right) Profiles

Model Tuning – 3

3. Network Weight Initialization
   - Optimal: **He_normal**

Training Accuracy (Left)/ Loss (Right) Profiles

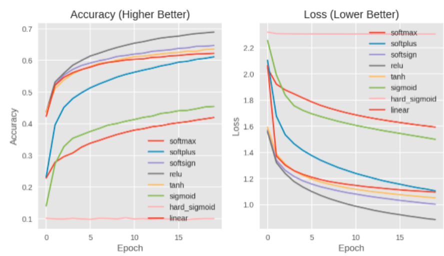


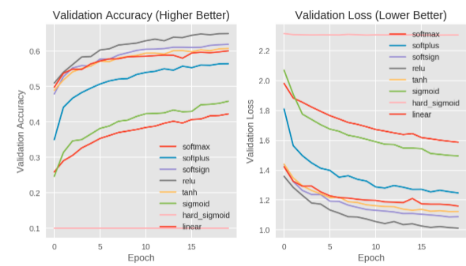


Testing Accuracy (Left)/ Loss (Right) Profiles

4. Neuron Activation Function
   - Optimal: **Relu**

Training Accuracy (Left)/ Loss (Right) Profiles

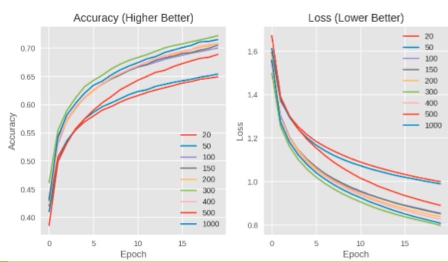


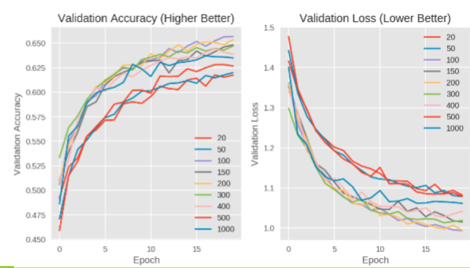


Testing Accuracy (Left)/ Loss (Right) Profiles

5. Number of Neurons in the Hidden Layer
   - Optimal: **200**

Training Accuracy (Left)/ Loss (Right) Profiles





Testing Accuracy (Left)/ Loss (Right) Profiles

Conclusion

- LeNet-5 Convolutional Neural Network was used to perform classification on CIFAR-10 dataset
  - Accuracy/Loss profiles used to evaluate performance with comparison to other models e.g. VGG-16, AlexNet
    - VGG-16 can achieve significant test accuracy/ loss improvements at the expense of much longer training times

- Regularization methods can be used to overcome overfitting issue
  - Dropout, L1/ L2 Regularization, Data Augmentation, Early Stopping
    - **Dropout** improves overfitting best

- Hyper-parameter tuning can be used for evaluation/ selection of the most optimal value for each parameter
  - Batch Size/ Training Optimization Algorithm/ Network Weight Initializer/ Neuron Activation Function/ Number of neurons in hidden layer
    - ~8% improvement in test accuracy can be achieved

References

- Keras Documentation
  - https://keras.io/
- Wrappers for the Scikit-Learn API
  - https://keras.io/scikit-learn-api/#wrappers-for-the-scikit-learn-api
- CIFAR-10 Dataset
  - https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
- *Gradient-Based Learning Applied to Document Recognition*, Yann Lecun, Leon Bottou, Yoshua Bengio and Patrick Haffner, 1998
  - http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf