THINKFUL

Analyzing Performance of Image Classification of CNN Models vs. Mobilenet_v2 Pre-trained Model

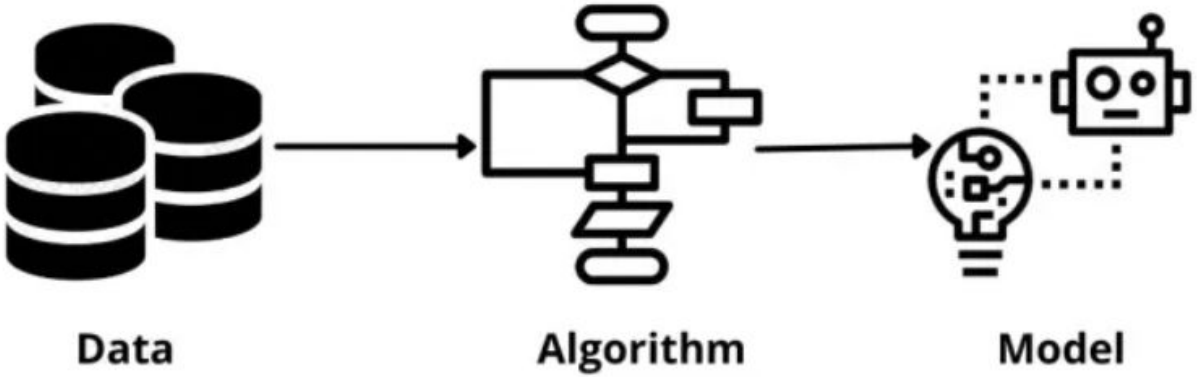
Malik Shavkatov

Table of Content

Background

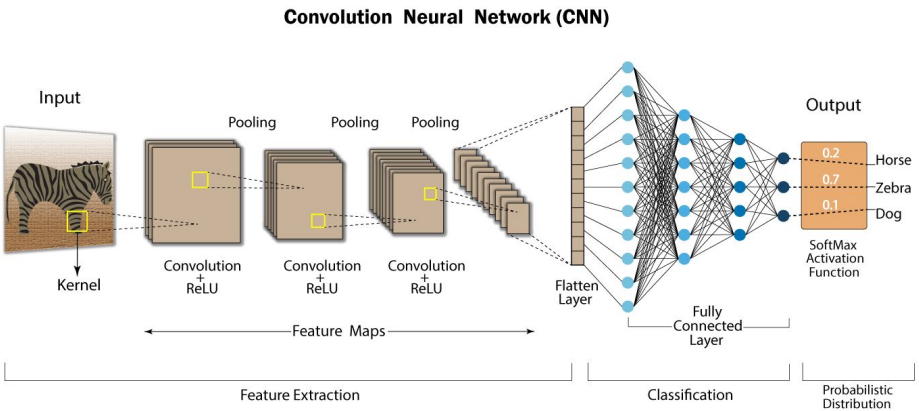
- Amount of new data being collected is increasing rapidly
- Building new ML algorithms to data is a challenge - costs money and time
    - > $100k and almost 3 months to build one ML model depending on complexity

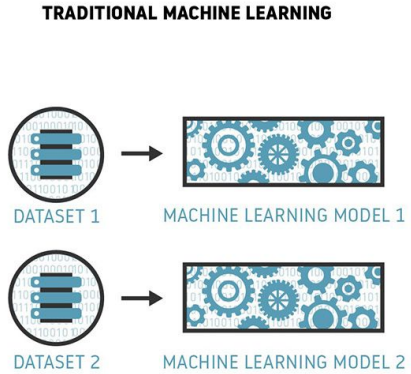
- Transfer Learning as an alternative method

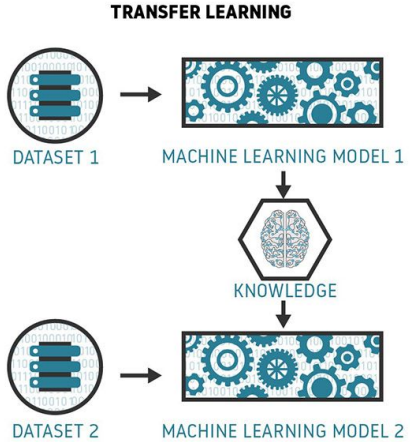


Introduction

- Convolutional Neural Networks (CNN) - deep learning algorithm for image analysis
- Transfer Learning - a pre-trained model (tf2-preview/mobilenet_v2/classification) from Tensorflow Hub







Objective

- This project will analyze the performance between a CNN model built from ground up vs. a pretrained model: Mobilenet_v2

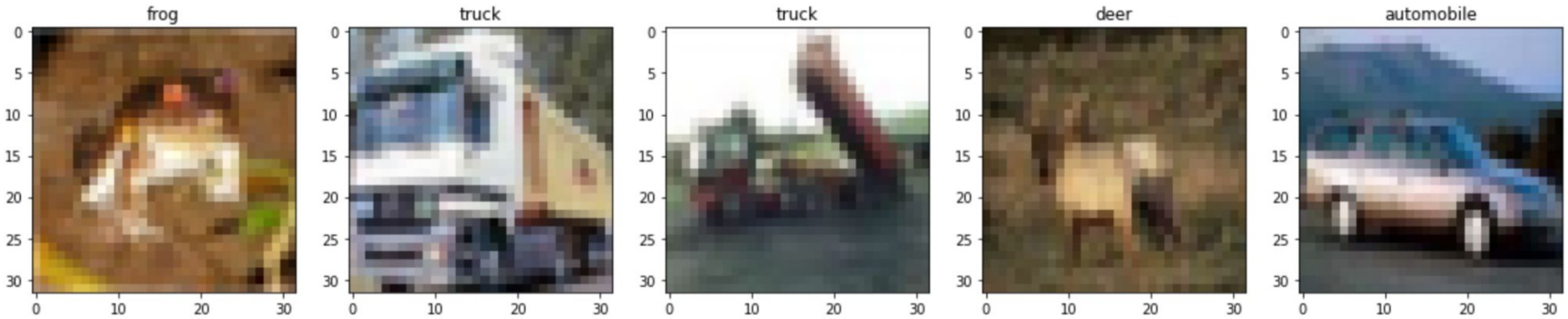
Why is this important?

- Building a model from ground up costs immense amount of money and time
- Implementing a pre-trained model to your dataset can potentially cut costs and also improve the performance of the final model

Data

- Cifar10 dataset consists of 60000 images with 32x32 pixels, divided into 10 classes
- Classes include: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck
- Images are converted to 224x224 pixels to fit into the pre-trained model
- First 10000 images are used from the training set and 2000 images from the test set
- Data can be accessed from https://www.cs.toronto.edu/~kriz/cifar.html

First 5 images from the training set



Methods

- Exploring and preprocessing the data
- Verify the data by checking image to class correspondence
- Resize and normalize data to feed into CNN
- Flatten the data to feed into classification layers
- Model 1 will be basic and evaluated to increase complexity for Model 2 and 3
- Tune hyperparameters to increase performance by each epoch
- Add regularization, bias, and dropout layers to decrease overfitting
- Implement Mobilenet_v2 pre-trained model as transfer learning technique
- Evaluate each model and compare

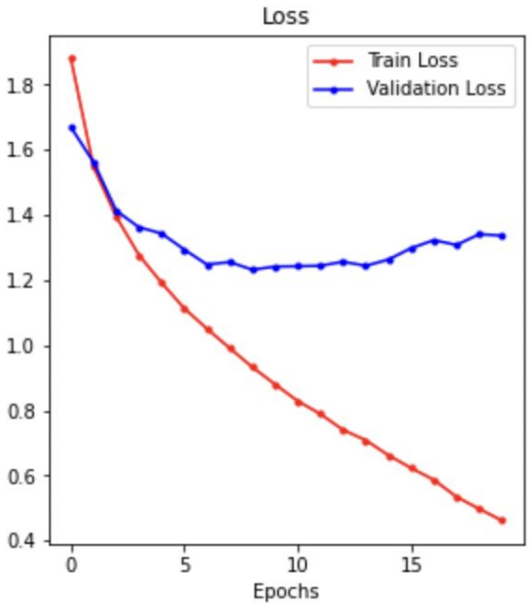
Results and Analysis

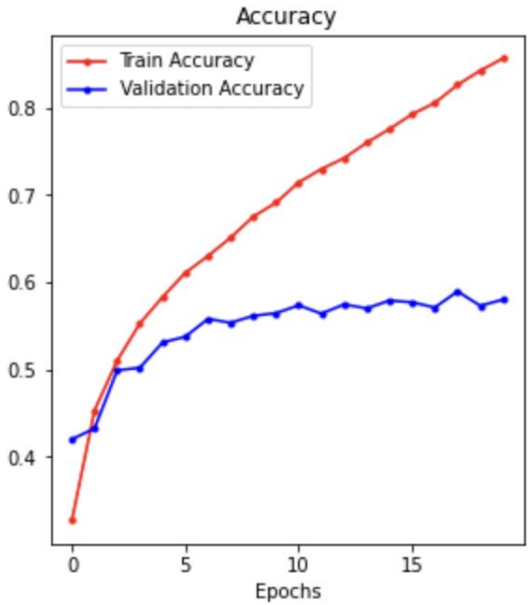
Model 1 - Basic CNN:

Conv2D>Maxpooling2D>Flatten>Dense

- 29s to run 20 epochs
- After 5 epochs, both loss and accuracy deviate from train and validation sets
- Model 1 overfits where final train accuracy is much higher than and validation accuracy

```
Loss:  1.336729407310485
Accuracy:  0.57999998331
Training time: 0:00:29.4
```



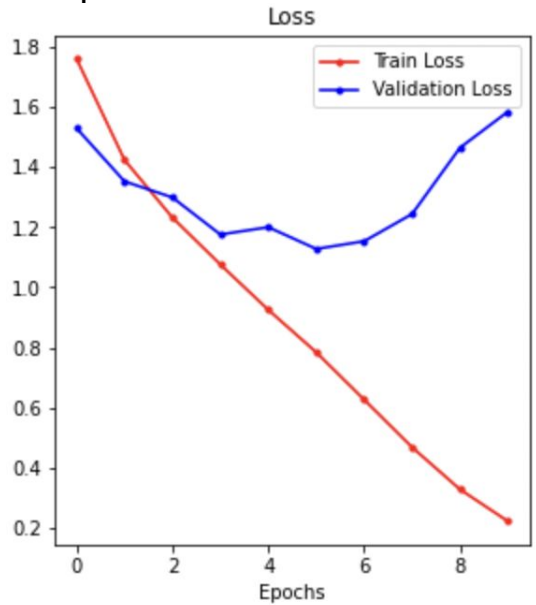


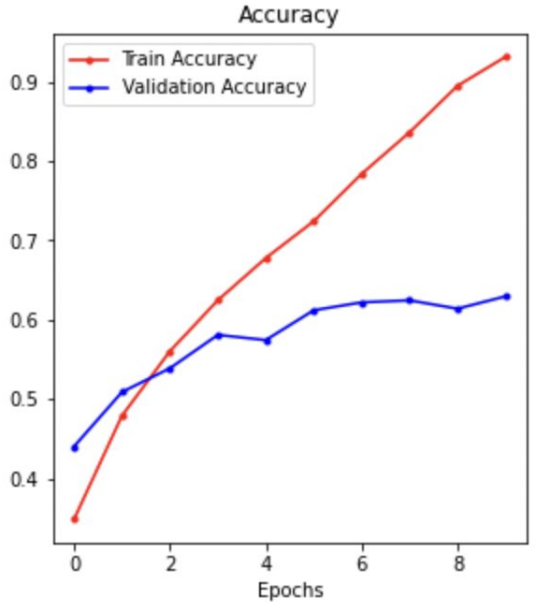
Model 2 - CNN:

Conv2D->Conv2D->Maxpooling2D->Conv2D->Maxpooling2d->Flatten->Dense
(Does include activation layers)

- 17s to run 10 epochs
- Performance increased by 5%
- Model validation loss increases after 7 epochs
- No change in validation accuracy after 5 epochs
- Model overfits

```
Loss:  1.585462450981140
Accuracy:  0.62949997186
Training time: 0:00:17.2
```



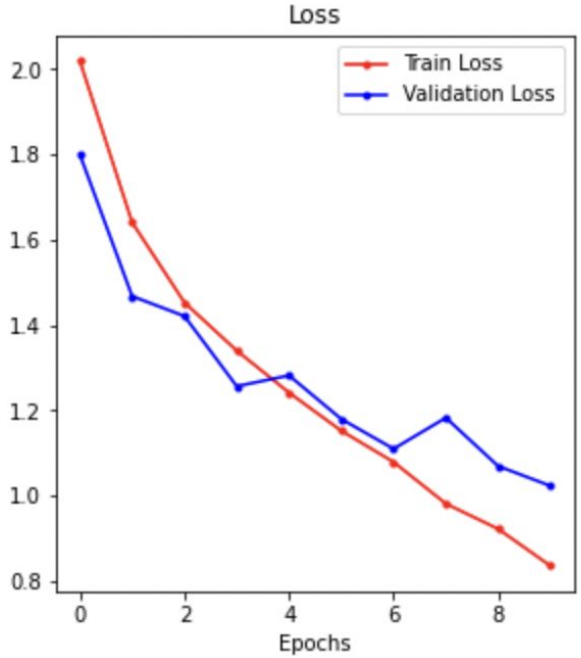


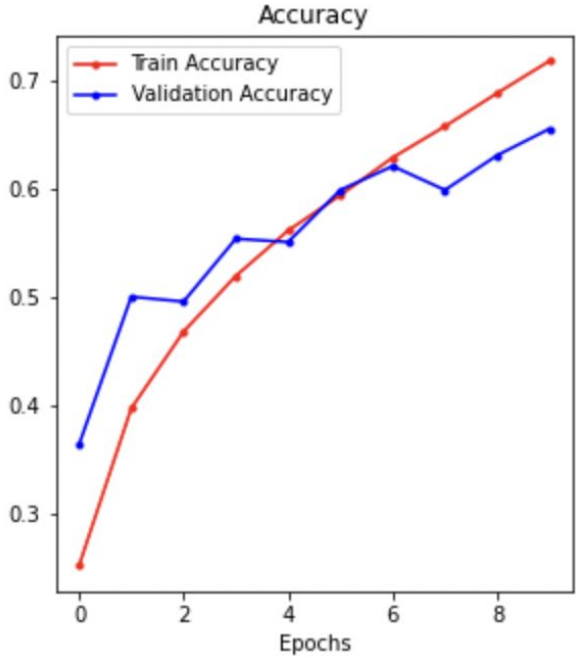
Model 3 - CNN: Adding L2 Regularization, Bias, and Dropout Layers to Reduce Overfitting

Conv2D->Conv2D->Maxpooling->Conv2D->Conv2D->Maxpooling->Dropout->Flatten->Dense->Dropout->Dense

- ~20s to run 10 epochs
- Validation loss deviates from train loss after 10 epochs
- Performance is increased by ~8%
- Reduced overfitting

```
Loss:  1.023271322250366
Accuracy:  0.65549999475
Training time: 0:00:19.5
```

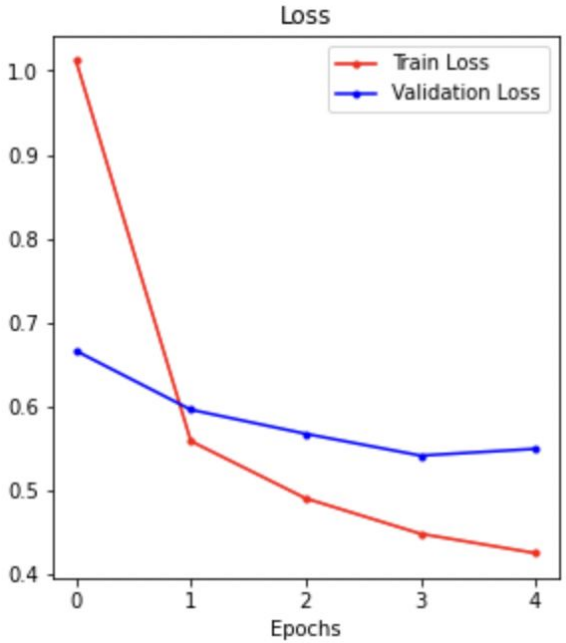


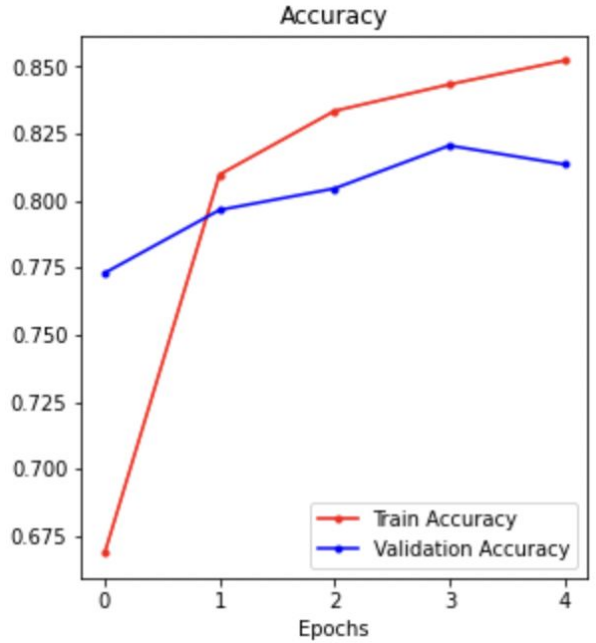


Model 4: Pre-Trained Model - Mobilenet_2v

- 90s to run 5 epochs
- Validation Loss and Accuracy stagnates after 2 epochs
- Overfitting is seen
- Performance increased by ~24%

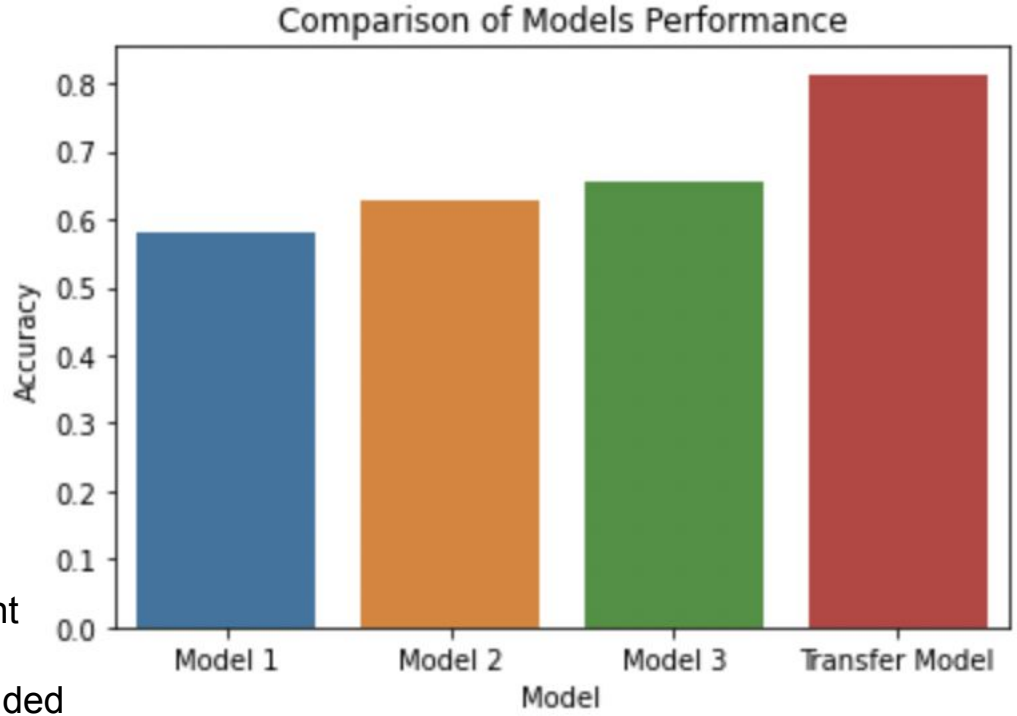
```
Loss:  0.548729956150054
Accuracy:  0.81349998712
Training time: 0:01:29.8
```





Model Comparison

	Model	Accuracy
0	Model 1	0.5800
1	Model 2	0.6295
2	Model 3	0.6555
3	Transfer Model	0.8135



- There is a trade-off between accuracy and processing time
- If processing time is a concern, model 3 might be better suited
- If not, model 4 - Transfer Model is recommended due to its higher performance

Final Thoughts and Recommendations

- Increasing complexity increases the accuracy and decreases the number of epochs to run to achieve good performance
- Use of regularization, dropout, and bias term reduced overfitting

How to Improve?

- Data size used was not large enough to train these CNN models, even though data was available.
- Increase computational efficiency
- Data augmentation and early stopping techniques to control overfitting
- GridSearchCV to find best hyperparameters

Questions?

Thank You!