ASSIGNMENT 4 REPORT

IJAZ MOHAMED UMAR

05.05.2022

Table of Contents

HOW TO RUN

The submission consists of trained models and the python notebook file for the assignment. The notebook is well documented to understand what each part of the code does and what and how the code works.

The food-101 dataset location is **very important**. If not already downloaded and stored in the working directory of assignment 4 folder, then the code will try to download the dataset. If you have a copy of the data set or will download it, then move that data set to the assignment 4 folder that will be submitted since the code uses the current working directory to load the data.

Make sure to run everything from the current working directory that has all the files in it such as if you're running from the Assignment 4 folder that is submitted then everything will be good, else you will need to adjust the file paths in the code that is given for it to run properly.

MODELS

1. Basic CNN Model. It consists of 8 layers. The first layer consists of an input image with dimensions of 224×224x3. It is convolved with 32 filters of size 5×5 resulting in a dimension of 112x112x32. The second layer is a MaxPooling operation which filters size 2×2 and stride of 2. Hence the resulting image dimension will be 56x56x32.

Similarly, the third layer also involves a convolution operation with 64 filters of size 3×3 followed by a fourth MaxPooling layer with similar filter size of 2×2 and stride of 2. Thus, the resulting image dimension will be reduced to 28x28x64.

Next, the fifth layer is the last convolutional layer of the Basic-CNN architecture and it has 128 filters of size 3×3 followed by a sixth GlobalAveragePooling layer that will get the average of each filter, and the resulting dimension will be (128,).

Once the image dimension is reduced, the seventh layer is a Dense layer with 256 units. In this layer, each of the 256 units in this layer will be connected to the (128,) units from the previous layers.

The final eighth layer will be a softmax output layer with 'n' possible classes depending upon the number of classes in the dataset. In our case, according to the food-101 dataset, n will be 101.

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 112, 112, 32)	2432
max_pooling2d (MaxPooling2D)	(None, 56, 56, 32)	0
conv2d_1 (Conv2D)	(None, 56, 56, 64)	18496
max_pooling2d_1 (MaxPooling2	(None, 28, 28, 64)	0
conv2d_2 (Conv2D)	(None, 28, 28, 128)	32896
global_average_pooling2d (Gl	(None, 128)	0
dense (Dense)	(None, 256)	33024
dense_1 (Dense)	(None, 101)	25957

Total params: 112,805
Trainable params: 112,805
Non-trainable params: 0

2. All CNN Model. It consists of seven layers. The first layer consists of an input image with dimensions of 224×224x3. It is convolved with 32 filters of size 5×5 resulting in a dimension of 112x112x32. The second layer is a MaxPooling operation which filters size 2×2 and stride of 2. Hence the resulting image dimension will be 56x56x32.

Similarly, the third layer also involves a convolution operation with 64 filters of size 3×3 followed by a fourth MaxPooling layer with similar filter size of 2×2 and stride of 2. Thus, the resulting image dimension will be reduced to 28x28x64.

Next, the fifth layer is the last convolutional layer of the Basic-CNN architecture and it has 128 filters of size 3×3 followed by a sixth GlobalAveragePooling layer that will get the average of each filter, and the resulting dimension will be (128,).

The final seventh layer will be a softmax output layer with 'n' possible classes depending upon the number of classes in the dataset. In our case, according to the food-101 dataset, n will be 101.

The only difference of **All-CNN Model** from **Basic-CNN** is that All-CNN model architecture doesn't have a dense layer with 256 units in between global average pooling and the last dense classification layer.

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 112, 112, 32)	2432
max_pooling2d (MaxPooling2D)	(None, 56, 56, 32)	0
conv2d_1 (Conv2D)	(None, 56, 56, 64)	18496
max_pooling2d_1 (MaxPooling2	(None, 28, 28, 64)	0
conv2d_2 (Conv2D)	(None, 28, 28, 128)	32896
global_average_pooling2d (Gl	(None, 128)	0
dense_1 (Dense)	(None, 101)	13029

Total params: 66,853
Trainable params: 66,853
Non-trainable params: 0

3. **Regularized CNN Model.** This Model is the same as the **Basic-CNN Model.** Only difference is that the Regularized CNN Model has three Dropout layers. The first Dropout layer is placed after the second MaxPooling layer and its drop rate is 20%. The second Dropout layer (20% drop rate) is located after the GlobalAveragePooling layer. And the last Dropout layer is placed before the output layer. The last Dropout layer's drop rate is 30%.

Model: "sequential"

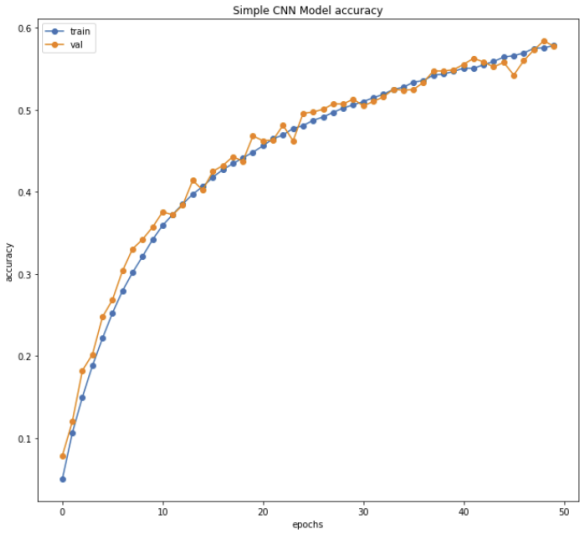
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 112, 112, 32)	2432
max_pooling2d (MaxPooling2D)	(None, 56, 56, 32)	0
conv2d_1 (Conv2D)	(None, 56, 56, 64)	18496
max_pooling2d_1 (MaxPooling2	(None, 28, 28, 64)	0
dropout (Dropout)	(None, 28, 28, 64)	0
conv2d_2 (Conv2D)	(None, 28, 28, 128)	32896
global_average_pooling2d (Gl	(None, 128)	0
dropout_1 (Dropout)	(None, 128)	0
dense (Dense)	(None, 256)	33024
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 101)	25957

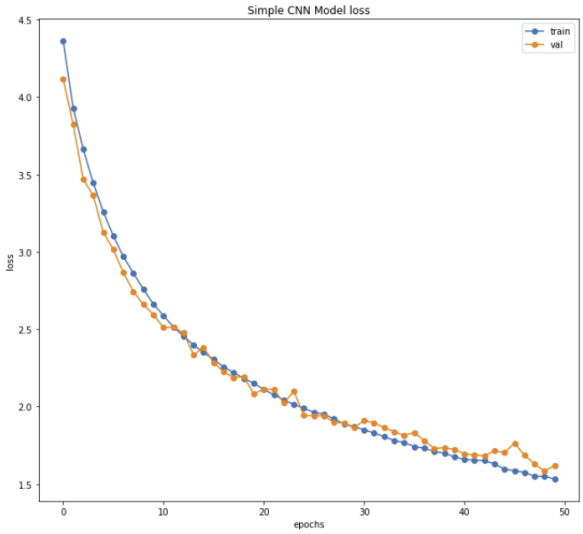
Total params: 112,805
Trainable params: 112,805
Non-trainable params: 0

4. **Transfer Learning Model.** The transfer learning solution is based on the **EfficientNetB0** pretrained model. An EfficientNetB0 model was trained on a large dataset called **ImageNet.** Mobile sized EfficientNetB0 architecture has about 4.2 million parameters. Input layer of EfficientNetB0 gets (224, 224, 3) size image and output layer provides (1280,) vector. Output layer connected with Global Average Pooling layer. Next, the Global Average Layer is connected with the Dropout layer with 30% drop rate, and finally with the Dense layer (101 units, 'softmax' activation function).

RESULTS

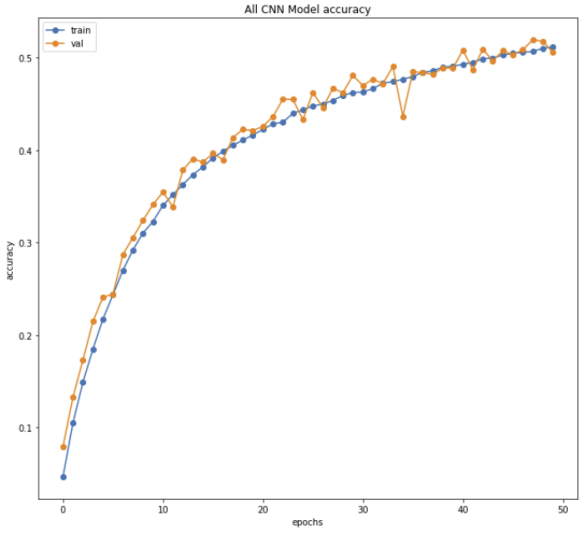
1. Basic-CNN Model Train-val Accracy & Loss

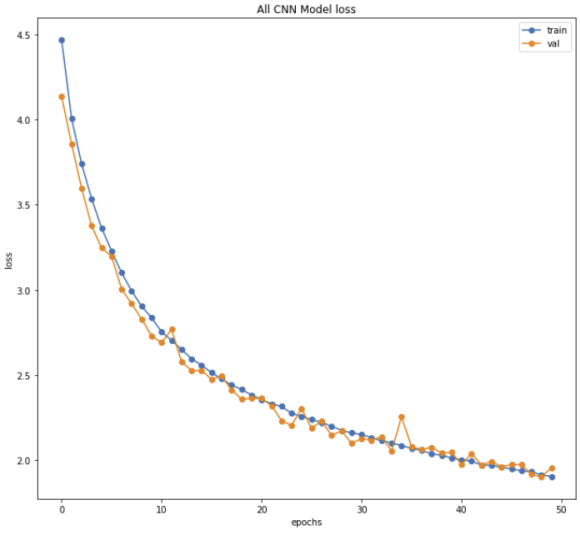




The **Basic-CNN** model was trained 50 epochs with the Food-101 dataset. The line plots above illustrate Train Val Accuracy and Loss. The train&val accuracy reached about 60%. However, test accuracy was around 39%. The Basic-CNN model overfitted with train& val data. There is almost a 20% difference between train&val and test accuracy.

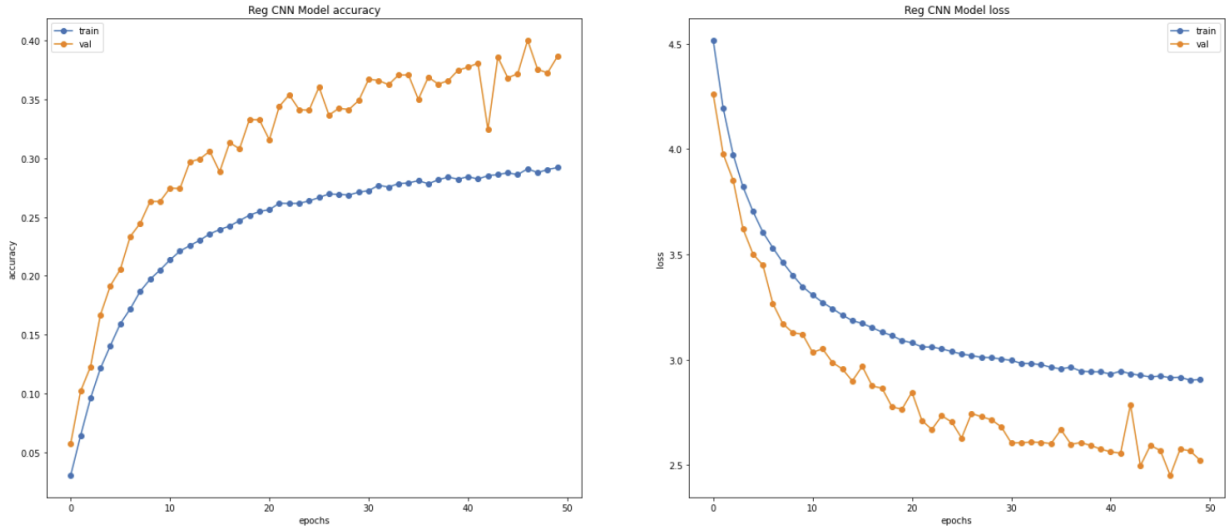
2. All-CNN Model Train-val Accuracy & Loss





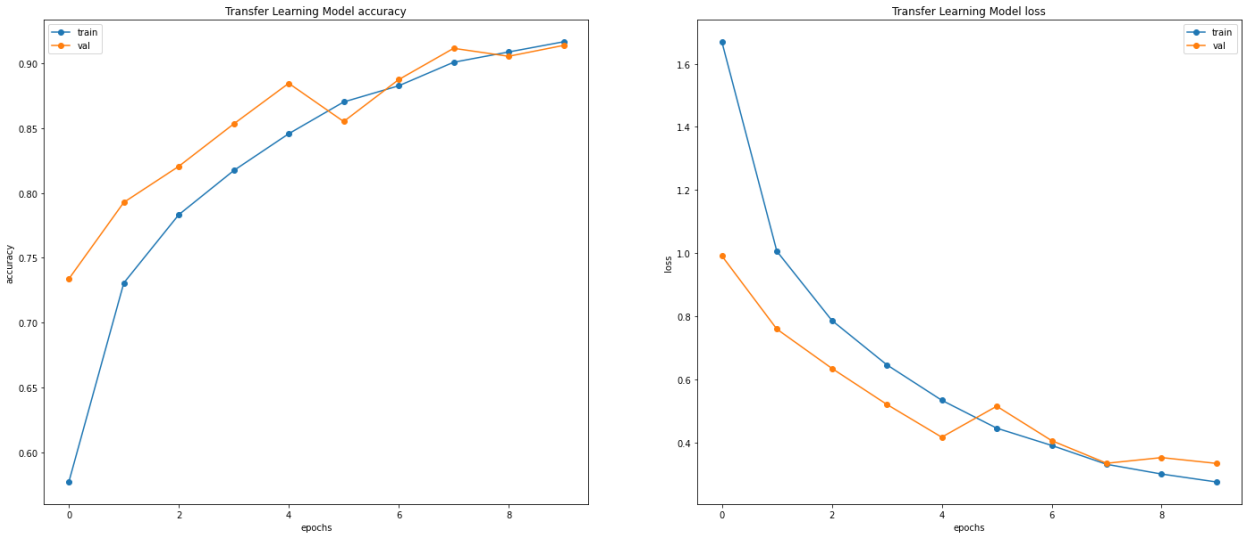
The **All-CNN** model was trained 50 epochs with the Food-101 dataset. The line plots above illustrate Train&Val Accuracy and Loss. The train&val accuracy reached about 50%. However, test accuracy was around 40%. This model's test accuracy is about 2% better than the previous one. The All-CNN model overfitted with train& val data. There is almost a 10% difference between train&val and test accuracies.

3. Regularized-CNN Model Train-val Accuracy & Loss



The **Regularized-CNN** model was trained 50 epochs with the Food-101 dataset. The line plots above illustrate Train&Val Accuracy and Loss. The highest train accuracy was about 28% and Val accuracy was around 45%. And, test accuracy was around 38%. This model's test accuracy is about 3% less than the All-CNN model. The model can reach higher accuracy if it is trained again.

4. Transfer Learning with EfficientNetB0 Model Train-Val Accuracy & Loss



The **Transfer Learning based** model was trained on 10 epochs with Food-101 dataset. The line plots above illustrate Train&Val Accuracy and Loss. The train&val accuracy reached about 90%. However, test accuracy was around 76%. This model's test accuracy is about 2 times better than previous models.

All Models Summary

#	Model	img_size	Number of parameters	Train/Val Size	Test Accuracy
1	Basic-CNN	(224,224,3)	112,805	70%-30%	**39.7%**
2	All-CNN	(224,224,3)	66,853	70%-30%	**41.1%**
3	Regularized-CNN	(224,224,3)	112,805	70%-30%	**38.2%**
4	Transfer Learning with EfficientNetB0(pretrained with ImageNet) model	(224,224,3)	4,178,952	70%-30%	**76.1%**

REFERENCES

1. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks
2. Food-101 – Mining Discriminative Components with Random Forests
3. Tensorflow tutorials
4. Public Image Classification ranking on Food-101 dataset
5. Optimization Algorithms in Deep Learning