**Assignment 6**

**Image Classification**

**Submitted By: Neel Vijay Patil (11436004)**

**Submitted To: Dr. Junhua Ding**

**INFO 5505 Applied Machine Learning for Data Science**

**UNT**

I have gathered the Intel Image classification dataset from Kaggle, and our main objective is to design the Model based on Convolutional Neural Network Image Classifier. The Dataset comprises of 25000 images of 150\*150 dimensions.

The Image dataset can be categorized into 6 different categories

* 'buildings'=0,
* 'forest'=1
* 'glacier'=2
* 'mountain'=3
* 'sea'=4
* 'street'=5

The Dataset consist of 3 different data test (3000), train (14000), predict (7000).

**Step 1: Importing Required libraries**

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Step 2: Created new API token and extracted the kaggle.json file from personal Kaggle account.**

Graphical user interface, text, application

Description automatically generated

**Step 3: Download the intel-image-classification Zip file to /content**

Graphical user interface, text, application, email

Description automatically generated

**Step 4: Unzip the intel -image-classification.zip file**

Application

Description automatically generated with medium confidence

**Step 5: Assigning the name of device and specified path in file**

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Initially before designing the CNN model. I have preprocessed the all the train, predict and test datasets. Here I have normalized the test and train datasets from 255 where I got the pixel value between 0 and 1. In order to train the dataset I have used the argumentation method.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Step 6: Performing the label encoding**

Graphical user interface, text, application

Description automatically generated

**Step 7: Defining functions for reading the image and displaying the image.**

Text

Description automatically generated

**Step 8: Finding the length of training and predicting dataset**

Graphical user interface, text, application, email

Description automatically generated

**Step 9: Designing the CNN Model where the images are categorized into 6 different categories using sequential model.**

The CNN layer comprises of following

**Layer 1:** The CNN input layer contains the image data. The data is in the form of 3D matrix and need to transform into the single column.

**Layer 2:** The convo is basically used to calculate the filter and receptive field dot products while the yield volume is addressed in form of the 1-valued integer. The main function the convo layer is to extract the image features hence its also known as the feature extraction layer which extracts the image features. The process is performed into loop where filter transfers on other receptive field which belongs to same image in input. To reduce the negative co-ordinates to 0 the Convo layer utilizes the ReLu Activation.

**Layer 3:** To reduce the spatial volume of the input image it uses the pooling layer. It is used to join the 2 layers all together.

**Layer 4:** The fully connected layer comprises of neurons, weights, and biases it is used for sorting the pictures into different categories.

**Layer 5:** The SoftMax layer also known as logistic layer is present at the bottom of the FC Layer in which binary classification is done using the logistic layer while multi classification is done using the SoftMax layer.

**Layer 6:** The output layer stores the one hot encoded label.

Graphical user interface, text, application, email

Description automatically generated

**Step 10: CNN Model Cross validation**

**n-splits = 10**

Here I have trained the model using the 10-fold cross validation. Every layer has stored accuracy score of validation and model in form of List. validation data constitutes ten percentage of the total data to be trained. If the loss of validation is not improved in consecutive epochs then the training of every layer is stopped, It displays the score at end.

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

-----------------------------------

Training on fold 1

Epoch 1/5

/usr/local/lib/python3.7/dist-packages/tensorflow/python/util/dispatch.py:1082: UserWarning: "`sparse\_categorical\_crossentropy` received `from\_logits=True`, but the `output` argument was produced by a sigmoid or softmax activation and thus does not represent logits. Was this intended?"

return dispatch\_target(\*args, \*\*kwargs)

395/395 [==============================] - 27s 38ms/step - loss: 1.2457 - accuracy: 0.4870 - val\_loss: 0.9844 - val\_accuracy: 0.6026

Epoch 2/5

395/395 [==============================] - 15s 38ms/step - loss: 0.9743 - accuracy: 0.6177 - val\_loss: 0.9978 - val\_accuracy: 0.6453

Epoch 3/5

395/395 [==============================] - 15s 39ms/step - loss: 0.8866 - accuracy: 0.6645 - val\_loss: 0.8789 - val\_accuracy: 0.6866

Epoch 4/5

395/395 [==============================] - 15s 38ms/step - loss: 0.8036 - accuracy: 0.7053 - val\_loss: 0.7648 - val\_accuracy: 0.7279

Epoch 5/5

395/395 [==============================] - 15s 38ms/step - loss: 0.7435 - accuracy: 0.7252 - val\_loss: 0.7547 - val\_accuracy: 0.7365

44/44 [==============================] - 0s 5ms/step - loss: 0.7547 - accuracy: 0.7365

Score of fold: 1: loss is 0.7546999454498291, accuracy is 0.7364672422409058

-----------------------------------

Training on fold 2

Epoch 1/5

395/395 [==============================] - 16s 39ms/step - loss: 1.1804 - accuracy: 0.5170 - val\_loss: 1.1215 - val\_accuracy: 0.5584

Epoch 2/5

395/395 [==============================] - 15s 38ms/step - loss: 0.9102 - accuracy: 0.6512 - val\_loss: 0.8699 - val\_accuracy: 0.6845

Epoch 3/5

395/395 [==============================] - 15s 38ms/step - loss: 0.8270 - accuracy: 0.6918 - val\_loss: 0.8170 - val\_accuracy: 0.7066

Epoch 4/5

395/395 [==============================] - 15s 38ms/step - loss: 0.7635 - accuracy: 0.7192 - val\_loss: 0.7702 - val\_accuracy: 0.7407

Epoch 5/5

395/395 [==============================] - 15s 38ms/step - loss: 0.6933 - accuracy: 0.7489 - val\_loss: 0.6989 - val\_accuracy: 0.7543

44/44 [==============================] - 0s 5ms/step - loss: 0.6989 - accuracy: 0.7543

Score of fold: 2: loss is 0.6989358067512512, accuracy is 0.754273533821106

-----------------------------------

Training on fold 3

Epoch 1/5

395/395 [==============================] - 16s 38ms/step - loss: 1.1955 - accuracy: 0.5243 - val\_loss: 0.9094 - val\_accuracy: 0.6489

Epoch 2/5

395/395 [==============================] - 15s 39ms/step - loss: 0.9037 - accuracy: 0.6584 - val\_loss: 0.8856 - val\_accuracy: 0.6417

Epoch 3/5

395/395 [==============================] - 15s 38ms/step - loss: 0.8310 - accuracy: 0.6888 - val\_loss: 0.7837 - val\_accuracy: 0.6937

Epoch 4/5

395/395 [==============================] - 15s 38ms/step - loss: 0.7601 - accuracy: 0.7205 - val\_loss: 0.7502 - val\_accuracy: 0.7194

Epoch 5/5

395/395 [==============================] - 15s 39ms/step - loss: 0.7126 - accuracy: 0.7413 - val\_loss: 0.7050 - val\_accuracy: 0.7229

44/44 [==============================] - 0s 5ms/step - loss: 0.7050 - accuracy: 0.7229

Score of fold: 3: loss is 0.7049649953842163, accuracy is 0.7229344844818115

-----------------------------------

Training on fold 4

Epoch 1/5

395/395 [==============================] - 16s 39ms/step - loss: 1.2664 - accuracy: 0.4981 - val\_loss: 0.9514 - val\_accuracy: 0.6246

Epoch 2/5

395/395 [==============================] - 15s 39ms/step - loss: 0.9684 - accuracy: 0.6257 - val\_loss: 0.9275 - val\_accuracy: 0.6446

Epoch 3/5

395/395 [==============================] - 15s 38ms/step - loss: 0.8574 - accuracy: 0.6828 - val\_loss: 0.8083 - val\_accuracy: 0.7101

Epoch 4/5

395/395 [==============================] - 15s 37ms/step - loss: 0.7691 - accuracy: 0.7191 - val\_loss: 0.7141 - val\_accuracy: 0.7407

Epoch 5/5

395/395 [==============================] - 15s 38ms/step - loss: 0.7176 - accuracy: 0.7385 - val\_loss: 0.8543 - val\_accuracy: 0.6838

44/44 [==============================] - 0s 5ms/step - loss: 0.8543 - accuracy: 0.6838

Score of fold: 4: loss is 0.8542846441268921, accuracy is 0.6837607026100159

-----------------------------------

Training on fold 5

Epoch 1/5

395/395 [==============================] - 16s 39ms/step - loss: 1.2047 - accuracy: 0.5118 - val\_loss: 0.9611 - val\_accuracy: 0.6237

Epoch 2/5

395/395 [==============================] - 15s 38ms/step - loss: 0.9247 - accuracy: 0.6517 - val\_loss: 0.9549 - val\_accuracy: 0.6536

Epoch 3/5

395/395 [==============================] - 15s 37ms/step - loss: 0.8160 - accuracy: 0.6919 - val\_loss: 0.7692 - val\_accuracy: 0.7177

Epoch 4/5

395/395 [==============================] - 15s 38ms/step - loss: 0.7691 - accuracy: 0.7149 - val\_loss: 0.8682 - val\_accuracy: 0.6536

Epoch 5/5

395/395 [==============================] - 15s 38ms/step - loss: 0.7219 - accuracy: 0.7359 - val\_loss: 0.6248 - val\_accuracy: 0.7684

44/44 [==============================] - 0s 5ms/step - loss: 0.6248 - accuracy: 0.7684

Score of fold: 5: loss is 0.6247870326042175, accuracy is 0.7683535218238831

-----------------------------------

Training on fold 6

Epoch 1/5

395/395 [==============================] - 16s 39ms/step - loss: 1.2034 - accuracy: 0.5181 - val\_loss: 0.9752 - val\_accuracy: 0.6329

Epoch 2/5

395/395 [==============================] - 15s 38ms/step - loss: 0.9231 - accuracy: 0.6498 - val\_loss: 0.8548 - val\_accuracy: 0.6892

Epoch 3/5

395/395 [==============================] - 15s 38ms/step - loss: 0.8084 - accuracy: 0.7011 - val\_loss: 0.8323 - val\_accuracy: 0.7014

Epoch 4/5

395/395 [==============================] - 15s 38ms/step - loss: 0.7556 - accuracy: 0.7292 - val\_loss: 0.7581 - val\_accuracy: 0.7477

Epoch 5/5

395/395 [==============================] - 15s 38ms/step - loss: 0.6979 - accuracy: 0.7447 - val\_loss: 0.6959 - val\_accuracy: 0.7584

44/44 [==============================] - 0s 5ms/step - loss: 0.6959 - accuracy: 0.7584

Score of fold: 6: loss is 0.6959352493286133, accuracy is 0.7583749294281006

-----------------------------------

Training on fold 7

Epoch 1/5

395/395 [==============================] - 16s 38ms/step - loss: 1.2450 - accuracy: 0.4929 - val\_loss: 1.0266 - val\_accuracy: 0.5852

Epoch 2/5

395/395 [==============================] - 15s 38ms/step - loss: 1.0113 - accuracy: 0.6073 - val\_loss: 0.9489 - val\_accuracy: 0.6500

Epoch 3/5

395/395 [==============================] - 15s 38ms/step - loss: 0.8829 - accuracy: 0.6673 - val\_loss: 0.7792 - val\_accuracy: 0.7156

Epoch 4/5

395/395 [==============================] - 15s 38ms/step - loss: 0.8066 - accuracy: 0.6992 - val\_loss: 0.7224 - val\_accuracy: 0.7484

Epoch 5/5

395/395 [==============================] - 15s 37ms/step - loss: 0.7437 - accuracy: 0.7258 - val\_loss: 0.6584 - val\_accuracy: 0.7698

44/44 [==============================] - 0s 5ms/step - loss: 0.6584 - accuracy: 0.7698

Score of fold: 7: loss is 0.6583611369132996, accuracy is 0.7697790265083313

-----------------------------------

Training on fold 8

Epoch 1/5

395/395 [==============================] - 15s 38ms/step - loss: 1.1926 - accuracy: 0.5241 - val\_loss: 0.9554 - val\_accuracy: 0.6044

Epoch 2/5

395/395 [==============================] - 15s 38ms/step - loss: 0.9214 - accuracy: 0.6490 - val\_loss: 0.8573 - val\_accuracy: 0.6764

Epoch 3/5

395/395 [==============================] - 15s 38ms/step - loss: 0.7986 - accuracy: 0.7044 - val\_loss: 0.7270 - val\_accuracy: 0.7398

Epoch 4/5

395/395 [==============================] - 15s 38ms/step - loss: 0.7241 - accuracy: 0.7325 - val\_loss: 0.7155 - val\_accuracy: 0.7391

Epoch 5/5

395/395 [==============================] - 15s 38ms/step - loss: 0.6904 - accuracy: 0.7473 - val\_loss: 0.6662 - val\_accuracy: 0.7598

44/44 [==============================] - 0s 5ms/step - loss: 0.6662 - accuracy: 0.7598

Score of fold: 8: loss is 0.6662044525146484, accuracy is 0.7598004341125488

-----------------------------------

Training on fold 9

Epoch 1/5

395/395 [==============================] - 16s 39ms/step - loss: 1.1774 - accuracy: 0.5341 - val\_loss: 0.9315 - val\_accuracy: 0.6436

Epoch 2/5

395/395 [==============================] - 15s 38ms/step - loss: 0.9093 - accuracy: 0.6538 - val\_loss: 0.8855 - val\_accuracy: 0.6892

Epoch 3/5

395/395 [==============================] - 15s 38ms/step - loss: 0.8217 - accuracy: 0.6948 - val\_loss: 0.7118 - val\_accuracy: 0.7356

Epoch 4/5

395/395 [==============================] - 15s 38ms/step - loss: 0.7458 - accuracy: 0.7242 - val\_loss: 0.6666 - val\_accuracy: 0.7676

Epoch 5/5

395/395 [==============================] - 15s 38ms/step - loss: 0.6959 - accuracy: 0.7430 - val\_loss: 0.6150 - val\_accuracy: 0.7698

44/44 [==============================] - 0s 5ms/step - loss: 0.6150 - accuracy: 0.7698

Score of fold: 9: loss is 0.6150299310684204, accuracy is 0.7697790265083313

-----------------------------------

Training on fold 10

Epoch 1/5

395/395 [==============================] - 16s 39ms/step - loss: 1.2375 - accuracy: 0.4951 - val\_loss: 0.9530 - val\_accuracy: 0.6115

Epoch 2/5

395/395 [==============================] - 15s 38ms/step - loss: 0.9669 - accuracy: 0.6267 - val\_loss: 0.9667 - val\_accuracy: 0.6201

Epoch 3/5

395/395 [==============================] - 15s 38ms/step - loss: 0.8685 - accuracy: 0.6715 - val\_loss: 0.7918 - val\_accuracy: 0.7156

Epoch 4/5

395/395 [==============================] - 15s 38ms/step - loss: 0.7575 - accuracy: 0.7203 - val\_loss: 0.7681 - val\_accuracy: 0.7406

Epoch 5/5

395/395 [==============================] - 15s 38ms/step - loss: 0.6968 - accuracy: 0.7438 - val\_loss: 0.6737 - val\_accuracy: 0.7541

44/44 [==============================] - 0s 5ms/step - loss: 0.6737 - accuracy: 0.7541

Score of fold: 10: loss is 0.6736576557159424, accuracy is 0.7540983557701111

K Fold accuracy and loss is obtained by calculating the average of the K\_Fold\_Accuracy and loss

Graphical user interface, text, application

Description automatically generated

**Step 11: Displaying the summary of the CNN.**

Graphical user interface

Description automatically generated with medium confidence

**Step 12: Plotting the line graph of Training and Validation Accuracy and Training and Validation Loss.**

Text

Description automatically generated

Chart, line chart

Description automatically generated

Chart, line chart

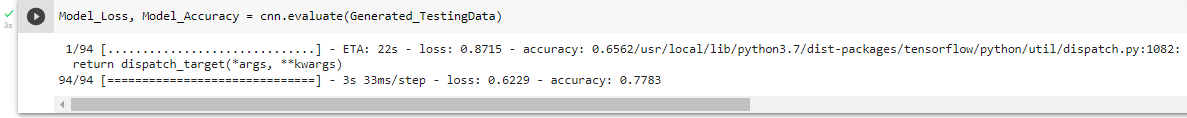
Description automatically generated

**Step 13: Fitting the model on the generated training data and defining the epochs as 5.**

Text

Description automatically generated with low confidence

**Step 14: Evaluating the model loss and accuracy for generated testing data.**



Graphical user interface, text, application

Description automatically generated

**Step 15: Printing the classification report**

Table

Description automatically generated

**Step 16: Plotting the Confusion Matrix on the testing data**

Text

Description automatically generated

Graphical user interface, application, Teams

Description automatically generated

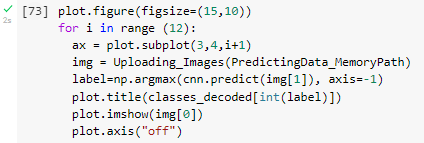
The diagonal row denotes the correctly predicted no of categories of test data from confusion matrix.

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated



A picture containing window, different

Description automatically generated

**Step 17: Importing Required Libraries**

Graphical user interface, text, application, email

Description automatically generated

**Step 18: Designing CNN Model which is based on Transfer Learning.**

Usually, huge amount of data is required to design the DL models which is costly and difficult to gather as well. Transfer learning can be solution such issues in which models trained on diversified and large datasets are applied to the problems.

Here I have selected the ResNet50V2 pretrained model because ImageNet data was trained and thus gained satisfactory outputs.

Preprocessing of Data: Data is divided in validation and training for evaluation of the model.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

**Step 19: Designing the ResNet50V2 CNN Model**

Various image classification issues are resolved by utilizing ResNet50V2 model which has huge collection of ImageNet dataset. The model uses the 150 layers. ResNet50V2 as opposed to being just heaped together, deep layers in the model are associated utilizing skip associations adding the first contribution to the convolution block's result. This guarantees unrivaled execution which makes it one of the most complicated NN.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Summary of the Base Conversion

Graphical user interface, text

Description automatically generated with medium confidence

Graphical user interface, text

Description automatically generated

Graphical user interface, website

Description automatically generated

Text

Description automatically generated

Fitting the tf\_model where value of epochs=5

A picture containing text

Description automatically generated

**Step 20: Calculating the model loss and accuracy**

Graphical user interface, text, application

Description automatically generated

**Step 21: Plotting the training and validation accuracy and loss.**

Text

Description automatically generated

A picture containing graphical user interface

Description automatically generated

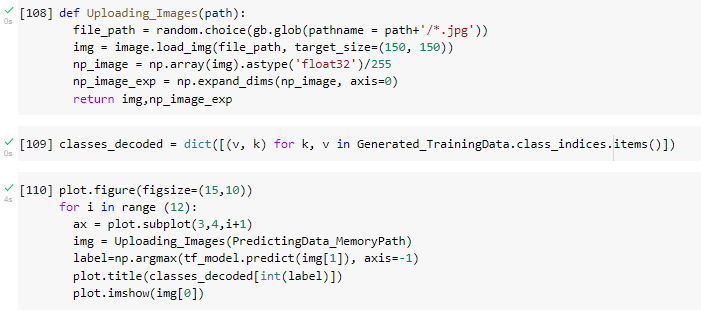
Chart, line chart

Description automatically generated

**Step 22: Generating the classification report**

Table

Description automatically generated



A picture containing window

Description automatically generated

Table

Description automatically generated

**Conclusion:** From the 2 models trained I found the model accuracy of CNN is 77 percentage while the transfer learning model has accuracy of 91 percentage. Hence the Transfer learning model is more accurate in predicting the images.