CS6910 - Assignment 2

Rishanth. R (CS18B044)
Rahul Chaurasia (CS20M002)

Rishanth R cs18b044

Number of later days used: 3.9 days

Link to this report:
https://wandb.ai/rishanthrajendhran/cs6910_assignment2/reports/CS6910-Assignment-2--Vmlldzo1OTc3ODk

PART A (Contributed by Rishanth .R (CS18B044))

Question 1

(a) What is the total number of computations done by your network? (assume mmm filters in each layer of size k×kk\times kk×k and nnn neurons in the dense layer)

Input dimensions = (256,256,3)

(Assuming appropriate padding for all layers)

Number of computations for 1st convolution : m * (256 * 256 * (k * k) * 3)

Number of computations for 1st maxpooling : m * (128 * 128 * (1) * m)

Number of computations for 2nd convolution : m * (128 * 128 * (k * k) * m)

Number of computations for 2nd maxpooling : m * (64 * 64 * (1) * m

Number of computations for 3rd convolution : m * (64 * 64 * (k * k) * m)

Number of computations for 3rd maxpooling : m * (32 * 32 * (1) * m)

Number of computations for 4th convolution : m * (32 * 32 * (k * k) * m)

Number of computations for 4th maxpooling : m * (16 * 16 * (1) * m)

Number of computations for 5th convolution : m * (16 * 16 * (k * k) * m)

Number of computations for 5th maxpooling : m * (8 * 8 * (1) * m)

Number of computations in the dense layer : m * 8 * 8 * n

Number of computations in output layer: n * 10

Softmax computation: 10 + 10 + 10 = 30

Total number of computations

= 16 * 16 * m * m * k * k * (75 * m + 256) + 331 * 8 * 8 * m * m + n * 10 + 30

(b) What is the total number of parameters in your network? (assume mmm filters in each layer of size k×kk\times kk×k and nnn neurons in the dense layer)

Each convolutional layer: m filters sof size k x k

Number of parameters in first convLayer = 3 * m * (k*k) (depth = 3 for RGB of input)

Number of parameters in subsequent convLayers = m * m * (k*k)

Total Number of parameters in convBlock

= 3 * m * (k*k) + (5-1) * m * m * (k*k)

= 3 * m * (k*k) + 4 * m * m * (k * k)

= (3 + 4 * m) * m * (k * k)

After 1st maxpooling (2 x 2 maxpooling) = 128 x 128 x m

After 2nd maxpooling (2 x 2 maxpooling) = 64 x 64 x m

After 3rd maxpooling (2 x 2 maxpooling) = 32 x 32 x m

After 4th maxpooling (2 x 2 maxpooling) = 16 x 16 x m

After 5th maxpooling (2 x 2 maxpooling) = 8 x 8 x m

Number of parameters in dense layer = m * 8 * 8 * n

Total number of parameters = m * ((3 + 4 * m) * (k * k) + 8 * 8 * n)

Question 2

To save time, the number of epochs was fixed at 5 for all models. 20 different combinations were tried out.

```
Parameters:
  numFilters:
    values: [16, 32, 64]
  numFiltersScheme:
    values: [0.5, 1, 2]
  datasetAugmentation:
    values: [False, True]
  batchNormalization:
    values: [False, True]
  dropout:
    values: [0.0, 0.25, 0.5]
  defaultActivationFunction:
    values: ["relu", "elu", "selu"]
```

numNeuronsInDenseLayers:

  values: [16, 32, 64]

filterSize:

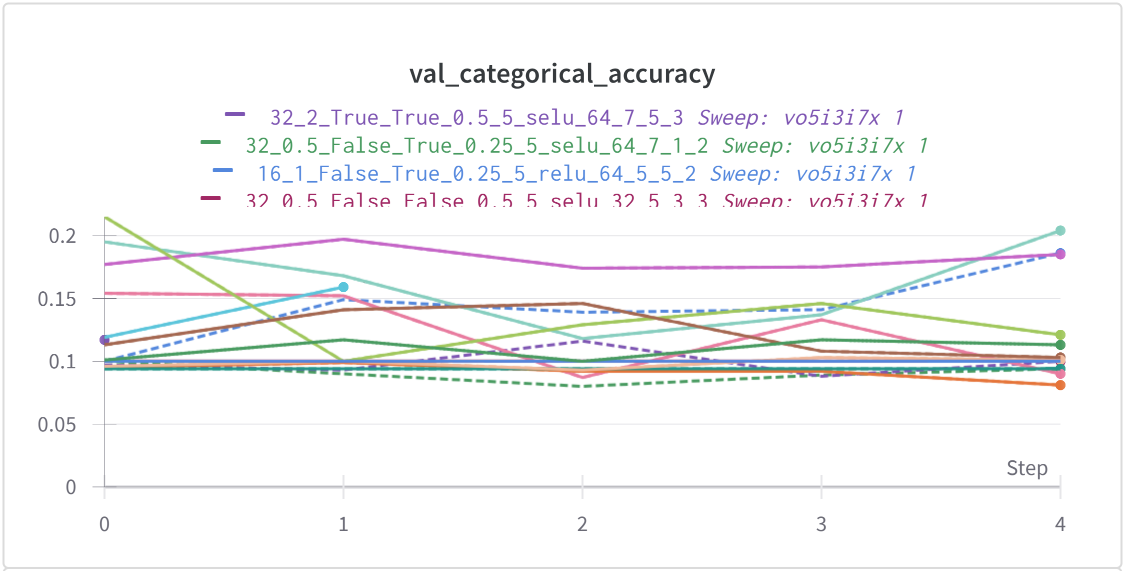
  values: [3, 5, 7]

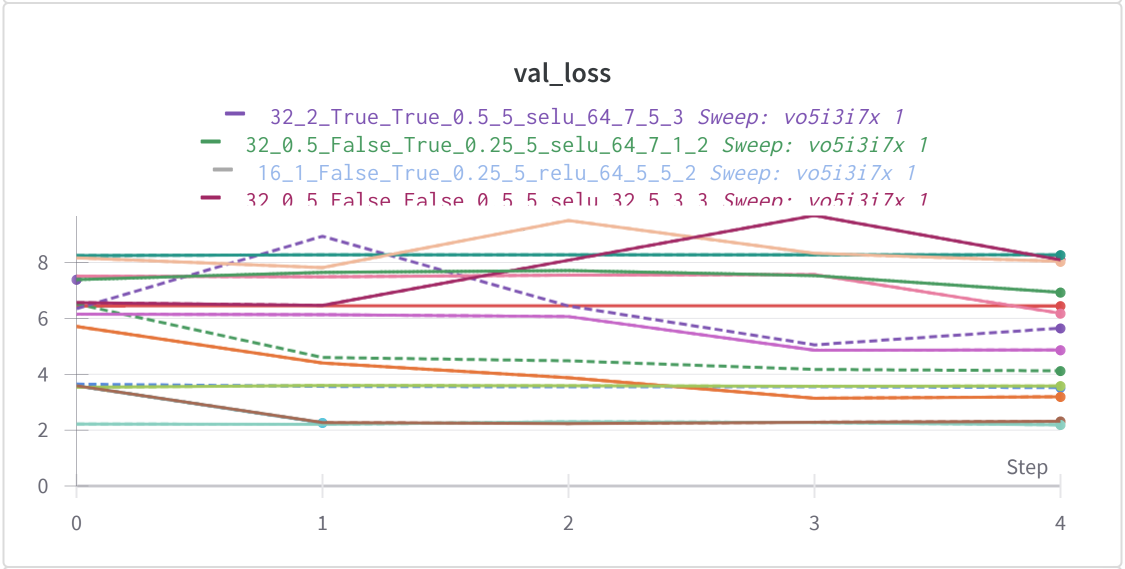
numConvBlocks:

  values: [1, 3, 5]

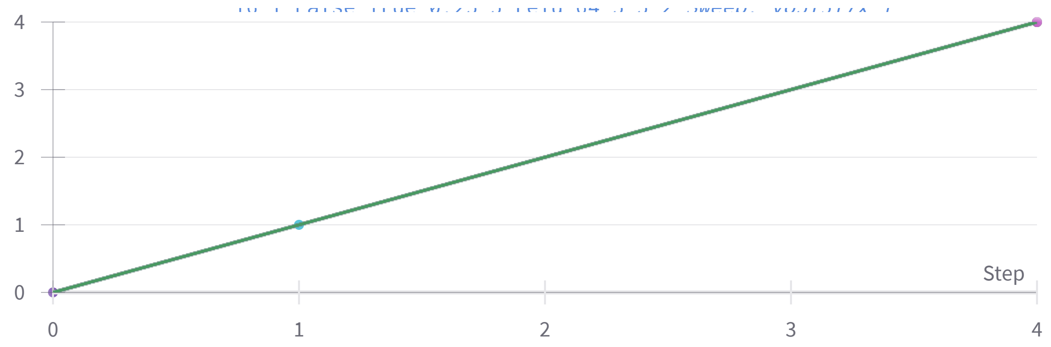
numDenseLayers:

  values: [1, 2, 3]

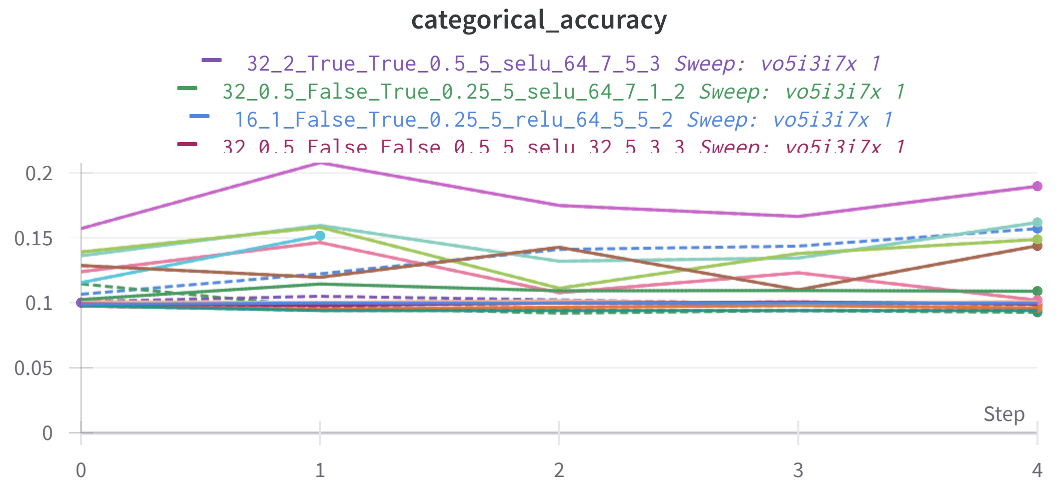




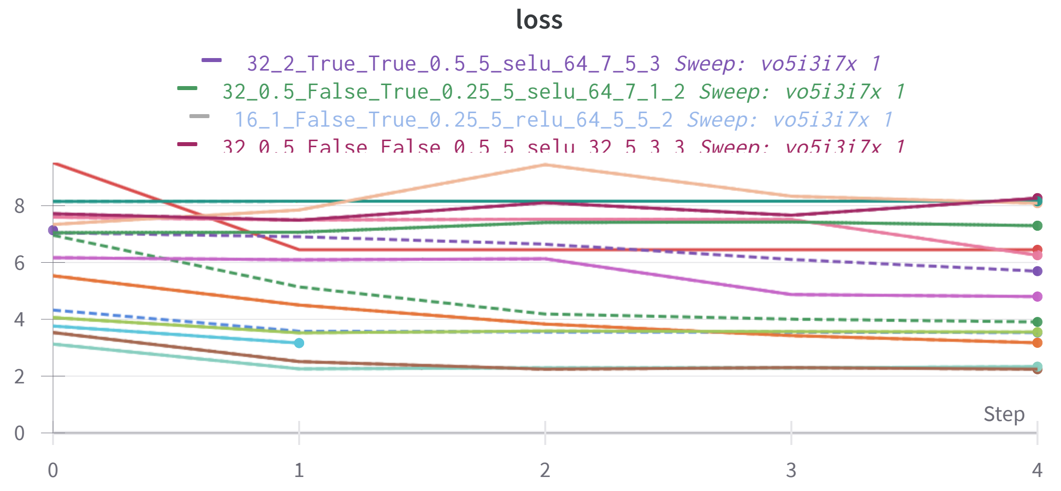
epoch
Showing first 10 runs
32_2_True_True_0.5_5_selu_64_7_5_3 Sweep: vo5i3i7x 1
32_0.5_False_True_0.25_5_selu_64_7_1_2 Sweep: vo5i3i7x 1
16_1_False_True_0.25_5_relu_64_5_5_2 Sweep: vo5i3i7x 1



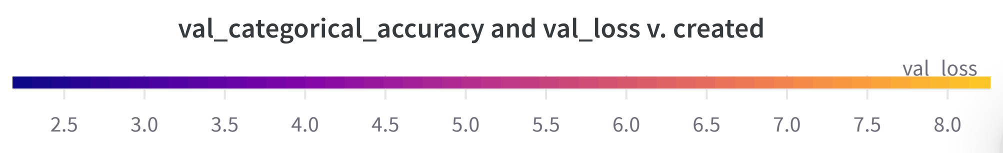
4
3
2
1
0
Step
0
1
2
3
4



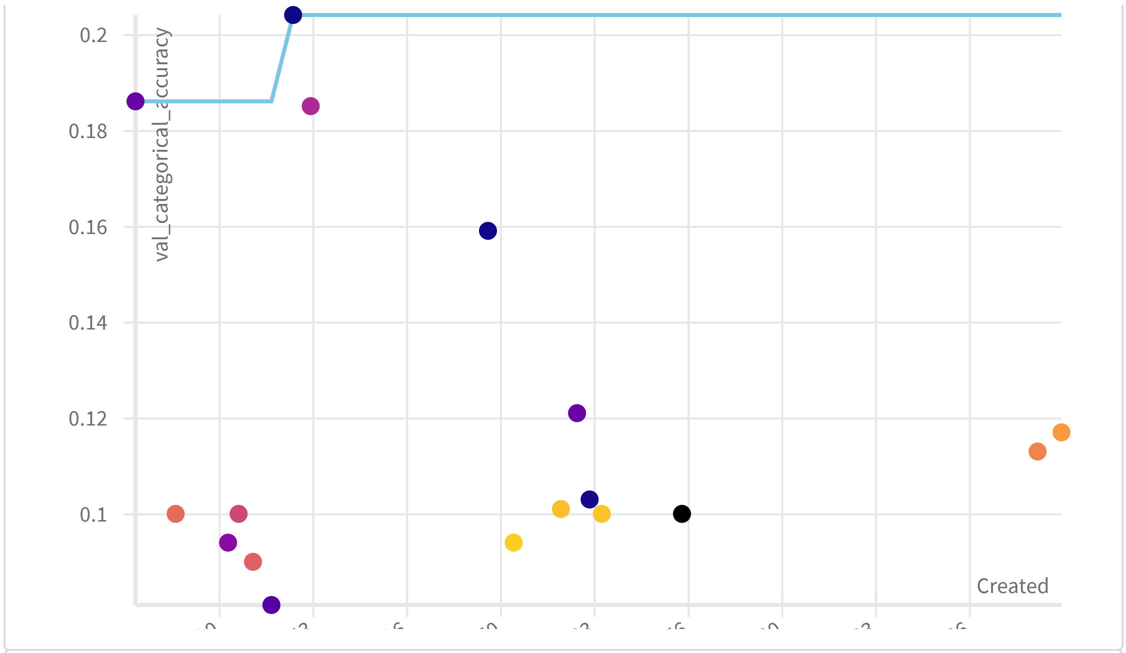
categorical_accuracy
32_2_True_True_0.5_5_selu_64_7_5_3 Sweep: vo5i3i7x 1
32_0.5_False_True_0.25_5_selu_64_7_1_2 Sweep: vo5i3i7x 1
16_1_False_True_0.25_5_relu_64_5_5_2 Sweep: vo5i3i7x 1
32_0.5_False_False_0.5_5_selu_32_5_3_3 Sweep: vo5i3i7x 1
0.2
0.15
0.1
0.05
0
Step
0
1
2
3
4

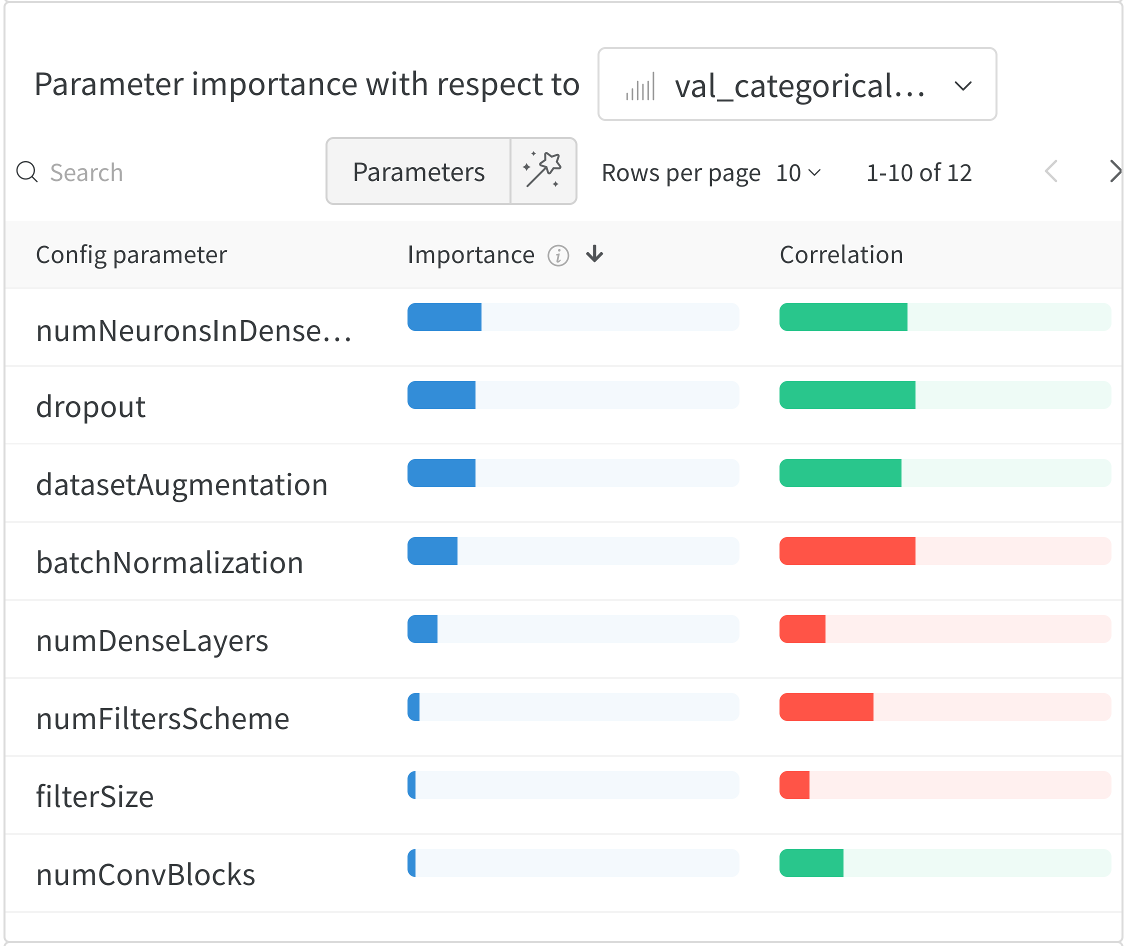


loss
32_2_True_True_0.5_5_selu_64_7_5_3 Sweep: vo5i3i7x 1
32_0.5_False_True_0.25_5_selu_64_7_1_2 Sweep: vo5i3i7x 1
16_1_False_True_0.25_5_relu_64_5_5_2 Sweep: vo5i3i7x 1
32_0.5_False_False_0.5_5_selu_32_5_3_3 Sweep: vo5i3i7x 1
8
6
4
2
0
Step
0
1
2
3
4



val_categorical_accuracy and val_loss v. created
val_loss
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

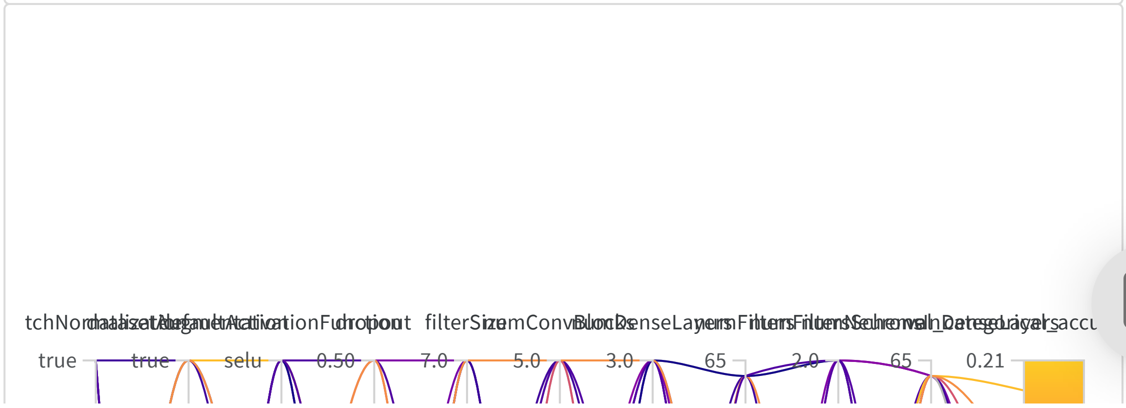


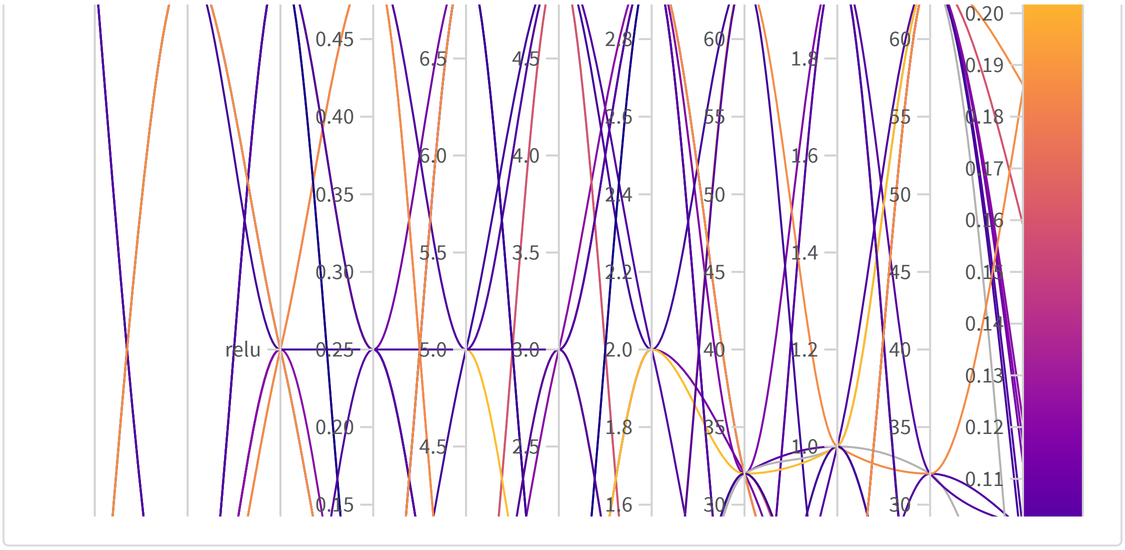


Parameter importance with respect to val_categorical…

Search | Parameters | Rows per page 10 | 1-10 of 12

Config parameter	Importance	Correlation
numNeuronsInDense…	🟦	🟩
dropout	🟦	🟩
datasetAugmentation	🟦	🟩
batchNormalization	🟦	🟥
numDenseLayers	🟦	🟥
numFiltersScheme	🟦	🟥
filterSize	🟦	🟥
numConvBlocks	🟦	🟩





Question 3

Increasing the number of neurons in the dense layers helped in learning more complicated patterns over the training set better and was able to give higher validation accuracies and lower losses.

Performing dataset augmentation on the training dataset is clearly helping generalize better over the training data without overfitting which is reflected in the lower validation losses when compared to models without dataset augmentation on the input data.

Surprisingly, batch normalization right after the activation layers seem to be lowering the validation accuracies. This could be because of doing it after activation and not before it. It may also be because the neural network wasn't deep enough to begin with added to the fact that we ran only for 5 epochs.

Increasing the number of dense layers and increasing the number of filters after each layer, both lead to an increase in validation loss which might be due to overfitting.

Smaller filter sizes gave better accuracies because they were able to capture the important data without considerable loss.

Larger number of convolutional layers also helped improve accuracies which might be because we are able to learn better patterns from the training data with more number of layers.

Question 4

Best Model:

numFilters = 32,

numFiltersScheme = 1.0,

datasetAugmentation = True,

batchNormalization = False,

dropout = 0.25,

numEpochs = 50,

defaultActivationFunction = "selu",

numNeuronsInDenseLayers = 64,

filterSize = 5,

numConvBlocks = 1,

numDenseLayers = 2,

inputDimensions = (256,256,3),

numOutputClasses = 10

(a)

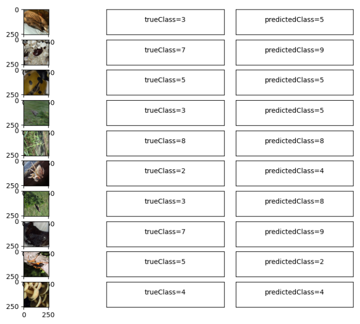
Categorical cross entropy loss: 2.0742404460906982

Categorical accuracy: 0.2752752900123596

This was achieved over the test set using the best model obtained after 20 epochs.

(b)

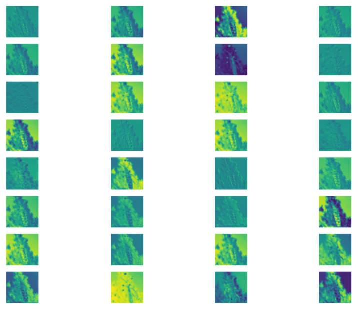
predictedClasses



```
trueClass=3    predictedClass=5
trueClass=7    predictedClass=9
trueClass=5    predictedClass=5
trueClass=3    predictedClass=5
trueClass=8    predictedClass=8
trueClass=2    predictedClass=4
trueClass=3    predictedClass=8
trueClass=7    predictedClass=9
trueClass=5    predictedClass=2
trueClass=4    predictedClass=4
```

(c)

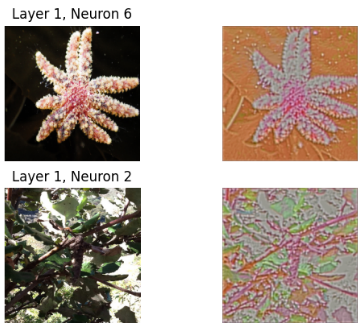
filtersVisualisation



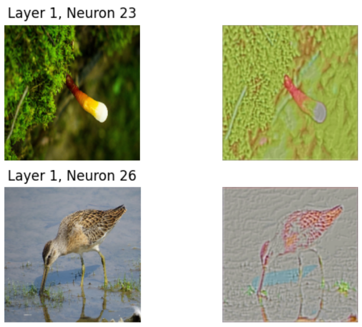
inputImage

Question 5

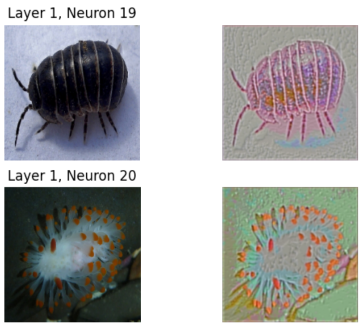
guidedBackProp4



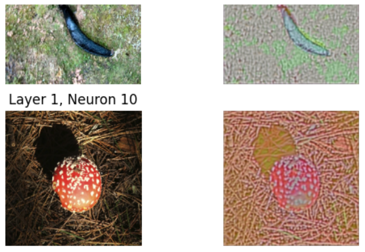
guidedBackProp0



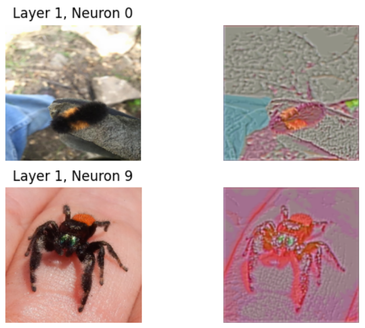
guidedBackProp1



guidedBackProp3



guidedBackProp2



Question 6:

Github Link:
https://github.com/RishanthRajendhran/cs6910_assignment2

****

PART B: (Contributed by Rahul Chaurasia (CS20M002))

Q1

(a) The dimensions of the images in your data may not be the same as that in the ImageNet data. How will you address this?

Ans. Typically we think of Convolutional Neural Networks as accepting fixed size inputs (i.e., 224×224,227×227, 299×299, etc). While we finetune a pretrained network on a custom dataset the input image dimensions may vary from the dimensions what the CNN was trained on. To address this issue, the dimensions of the input images of the custom dataset are changed to match the dimension requiremnets of the pretrained CNN model.

(b) ImageNet has 1000 classes and hence the last layer of the pre-trained model would have 1000 nodes. However, the naturalist dataset has only 10 classes. How will you address this?

Ans. To address the issue of difference in classes in the inaturalist dataset and the no of nodes in the fully connected layer of the pretrained CNN model, we remove the Fully connected layer of the pretrained CNN model and replace it with a fresh layer comprising 10 nodes ie equal to the number of classes in the inaturalist dataset. Thereafter, the model is finetuned and used for classifying the custom dataset.

Q2

You will notice that InceptionV3, InceptionResNetV2, ResNet50, Xception are very huge models as compared to the simple model that you implemented in Part A. Even fine-tuning on a small training data may be very expensive.

What is a common trick used to keep the training tractable (you will have to read up a bit on this)? Try different variants of this trick and fine-tune the model using the iNaturalist dataset. Forexample, '___'ing all layers except the last layer, '___'ing upto k layers and '___'ing the rest. Read

up on pre-training and fine-tuning to understand what exactly these terms mean.

Write down the different strategies that you tried (simple bullet points would be fine).

Ans. Strategies for Fine tuningare as under:

(a) Linear SVM on top of bottleneck features -The best strategy for a smaller dataset is

to train an SVM on top of the output of the convolutional layers just before the fully connected layers(also called bottleneck features).

(b) Just Replace and train the last layer:

ImageNet pretrained models have 1000 outputs from last layer, we can replace this with our

own softmax layers, for example in order to build 10 class classifier our softmax layer will have 10 output classes. Now, the back-propagation is run to train the new weights. In this case, however, it's likely for us to overfit so a lot of data augmentation and proper cross-validation is important.

(c) Train only last few layers: Depending on the amount of data available , the complexity of the problem , one can choose to freeze(ie not changing weights during backpropagation)first few layers and train only last few layers. The initial layers of convolutional neural networks just learn the general features like edges and very general image features, it's the deeper part of the networks that learn the specific shapes and parts of objects which are trained in this method. Another similar method is to use 0 or very small learning rate during the initial layers and using higher learning rate for the layers that are deeper.

(d) Freeze, Pre-train and Finetune(FPT): It is one of the most effective techniques for finetuning a pretrained model. There are two steps involved here:

i) Freeze and Pretrain: First replace the last layer with a small mini network of 2 small Fully connected layers. Now, freeze all the pretrained layers and train the new network. Save the weights of this network

ii) Finetune: Load the pretrained weights and train the complete network with a smaller learning rate. This results in very good accuracy with even small datasets.

(e) Train all the layers: In case of availability of a large data set(approx a million images) for training, we can start with pretrained weights but train the complete network.

Q3

Now finetune the model using different strategies that you discussed above and different hyperparameter choices. Based on these experiments write down some insightful inferences (once again you will find the sweep function to be useful to plot and compare different choices).

Here are some examples of inferences that you can draw:

• Using a huge pre-trained network works better than training a smaller network from scratch

(as you did in Part A)

• InceptionV3 works better for this task than ResNet50

• Using a pre-trained model, leads to faster convergence as opposed to training a model from scratch

• … ….

(Note: I don't know if any of the above statements is true. I just wrote some random comments that came to my mind) Of course, provide evidence (in the form of plots) for each inference.

Of course, provide appropriate plots for the above inferences (mostly automatically generated by wandb). The more insightful and thorough your inferences and the better the supporting evidence (in terms of plots), the more you will score in this question.

Q4

Paste a link to your github code for Part A

Example: https://github.com/<user-id>/cs6910_assignment2/partB

We will check for coding style, clarity in using functions and a README file with clear instructions on training and evaluating the model (the 10 marks will be based on this).

• We will also run a plagiarism check to ensure that the code is not copied (0 marks in the assignment if we find that the code is plagiarised).

• We will check the number of commits made by the two team members and then give marks accordingly. For example, if we see 70% of the commits were made by one team member

then that member will get more marks in the assignment (note that this contribution will

decide the marks split for the entire assignment and not just this question).

• We will also check if the training and test data has been split properly and randomly. You

will get 0 marks on the assignment if we find any cheating (e.g., adding test data to training

data) to get higher accuracy.

Ans. I Will share my google colab notebook comprising whatever i have been able to achieve till now.

The following link can be accessed from iitm smail accounts only.

Link : https://colab.research.google.com/drive/1Bu4FOIHlMkat7aE053tOIlIdnf755ymx?usp=sharing

****

PART C: (Contributed by Rishanth .R (CS18B044))

A video of a railway policeman saving the life of a passenger who fell into the gap between the train and the platform in Mumbai had been circulating in social media a while ago. While such incidents does reinforce faith in miracles, it also highlights the need to have a system in place to provide timely help in such scenarios. It might not always be the case that a railway policeman is around to save the day!

Accidents on railway tracks are common, especially in a country lik India with a very large railways network. Using object detection on footage from railway stations and railway tracks, we can identify people and animals stuck in the tracks. The system can then determine if there is a possibility of accident. If yes, the train operator can be informed about the same and appropriate action can be taken.

Youtube Link: youtube.com/watch?v=Pe_f6PbQlQQ

****
****

Self Declaration:

Contributions to the project:

Rishanth. R (CS18B044) - Part A and Part C

Rahul Chaurasia (CS20M002) - Part B

We, Rishanth .R and Rahul Chaurasia, swear on our honour that the above declaration is correct.

****
****

https://wandb.ai/rishanthrajendhran/cs6910_assignment2/reports/CS6910-Assignment-2--Vmlldzo1OTc3ODk