Spring 2021

A Study on Multi-Class Classification by Neural Networks

EE267 Project

Rojin Zandi

Contents:

1. Dataset Description

Intel image dataset [1] contains three files: training, test and prediction. In this project we are going to use the training and test set. Each one of the sets contains six different files which are the labels(classes) for the images. These labels are Buildings, Forest, Glacier, Mountain, Street, and Sea.

	Buildings	Forest	Glacier	Mountain	Sea	Street	Total
Training Set	2191	2271	2404	2512	2274	2382	14034
Test Set	437	474	553	525	510	501	3000

Table 1: Number of images in each class of the training and test set.

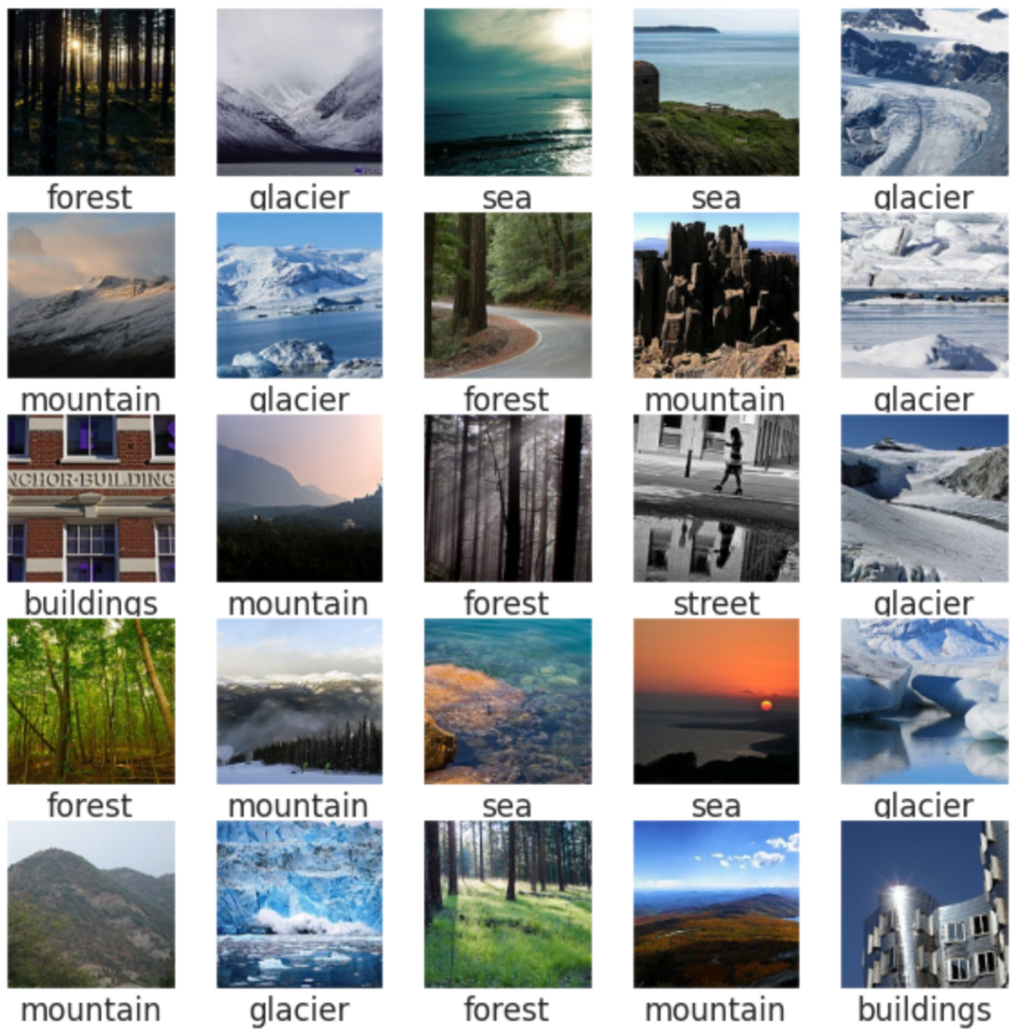

Fig.3: Some of images in the dataset

As it can be seen in Fig.1 and Fig.2, Mountain and Buildings classes has the most and least number of training data, respectively. On the test set, Glacier with 553 images is the largest and Buildings with 437 images is the smallest class of data [Table 1].

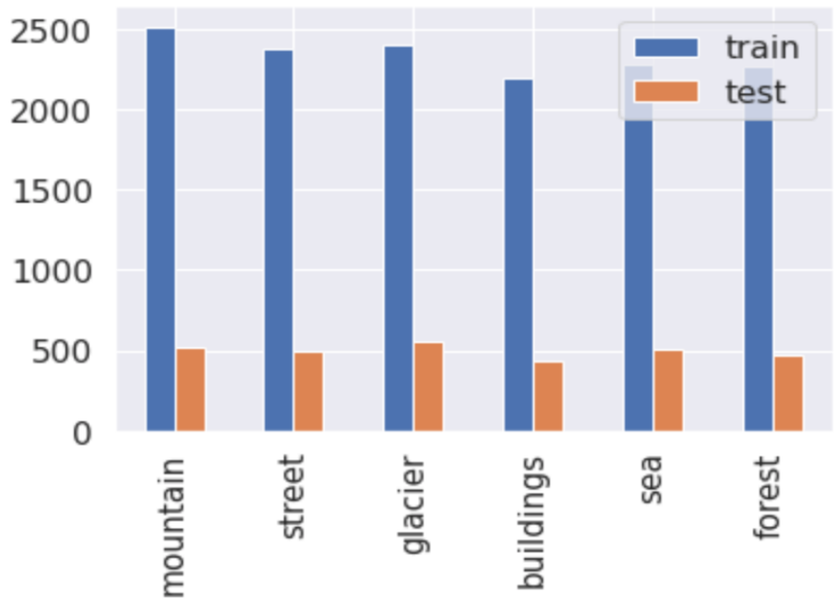

Fig. 1: Bar plot of the number of images in each class of the training and test set.

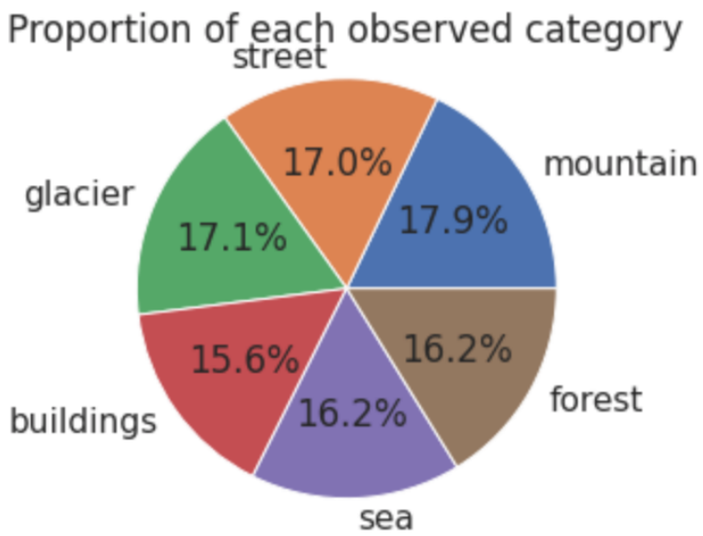

Fig. 2: Pie chart of the proportion of each observed category in the training set

2. Preprocessing

In Computer Vision, we need to provide clean and suitable data for feeding the algorithm. Preprocessing in deep learning is the process of transforming the raw data to a proper type which improves the accuracy of our models [2]. Preprocessing includes data cleaning, transformation, reduction, integration which depends on our data and application. For example, in Intel image dataset we were successful to use all the data and data reduction was not necessary, but the type of data needed to be transformed. The raw data is RGB image which is not suitable for feeding our neural network models. So, we have implemented a function in our code, which takes some information about the data, such as data path, images,

name of classes (labels), size of images. Notice that this function uses OpenCV library which the RGB images in this package are read BGR, so we need to change it. Finally, after resizing and calling the data, we have changed the type of images to float32 and labels to int32. After implementing the function, we can load the data from our drive and start data analysis.

3. MLP Models

In this project, we have implemented 11 Multi-Layer Perceptron models [3]. In each model we have changed one parameter which affect the efficiency and may decrease or increase model accuracy and loss. The training and test accuracy of all utilized models are provided in Table 2.
In the provided code, we have used Sequential API which allows us to create our models layer by layer, but we only can use it when the model is not sharing layers. Considering our data which is image, we need to use Flatten API to convert our input to a one-dimensional array. There is also Dense layer in our code, which connects every input to every output. In other words, it fully connects the neurons of the previous layer to the neurons of the next layer.

	# Layers	Optim. Method	Learning Rate	# Epochs	Activation function	Regul.	Val. Split	Training Accuracy	Test Accuracy
1	3	SGD	0.1	10	same	No	0.1	66.72%	47.83%
1.2	4	SGD	0.1	10	same	No	0.1	65.86%	53.53%
2	3	SGD	0.001	10	same	No	0.1	29.65%	28.2%
2.1	3	Adam	0.001	10	same	No	0.1	68.7%	57.56%
3	4	Adam	0.001	10	same	No	0.2	67.92%	58.93%
4	4	Adam	0.001	10	Tanh*	No	0.2	67.28%	57.46%
5	4	Adam	0.001	15	same	No	0.2	72.86%	60.93%
6	4	Adam	0.001	15	same	L1	0.2	21.89%	17.49%
6.1	4	Adam	0.001	15	same	L2	0.2	33.65%	33.43%
6.2	4	Adam	0.001	15	same	L1.L2	0.2	31.43%	22.83%
6.3	4	Adam	0.001	15	same	Dropout	0.2	69.72%	58.17%

Table 2: Parameters and accuracy of MLP models

3.1 Model Improvements

As discussed in section 3, changing the parameters such as number of layers or learning, can affect the accuracy and loss of the model. To compare the efficiency of different models, we have provided the test and training accuracies in Table 2 and also confusion matrix of each model (Fig. 4 to Fig. 14). We have used heat map which shows the highest values by lighter colors, so in our figures we prefer to have a lighter diagonal which means there are more correctly predicted data.

- **Model 1**
  In model 1, we have three layers which are two hidden layers, and output with 512, 128, and 6 neurons, respectively. The activation function in hidden layers is Rectified Linear Unit (ReLU) and in output SoftMax has been used. There are 10 epochs and the applied optimization method is Stochastic Gradient Descent (SGD) with 0.1 as learning parameter. We have also applied cross validation with proportion of 0.1.

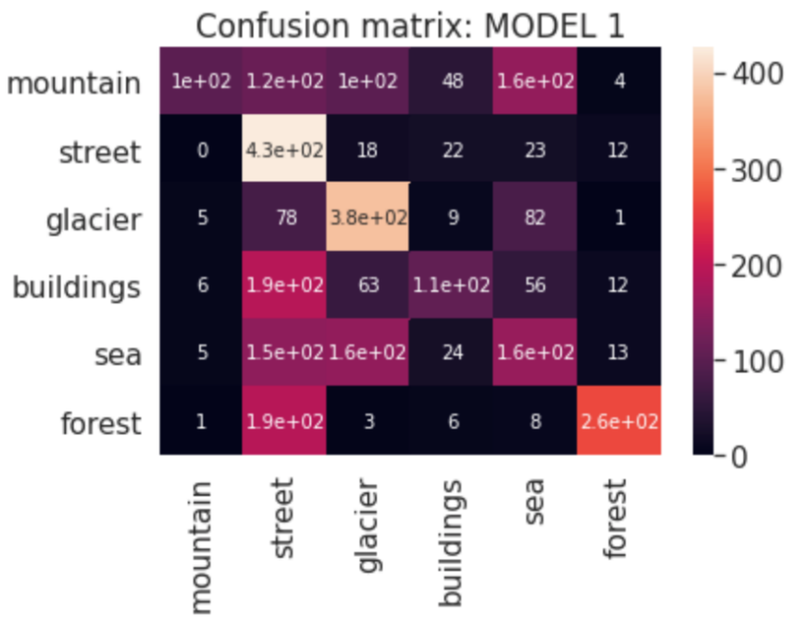

Fig. 4: Confusion matrix of model 1

	Mountain	Street	Glacier	Buildings	Sea	Forest
F1-score	0.31	0.51	0.59	0.33	0.32	0.67
Recall	0.19	0.85	0.68	0.24	0.31	0.55

Table 3: Recall of model 1

- **Model 1.2**
  Model 1.2 adds one hidden layer with same activation function (ReLU). So, the new model has three hidden layers and the test accuracy increases to 53.53%, comparing to model 1.

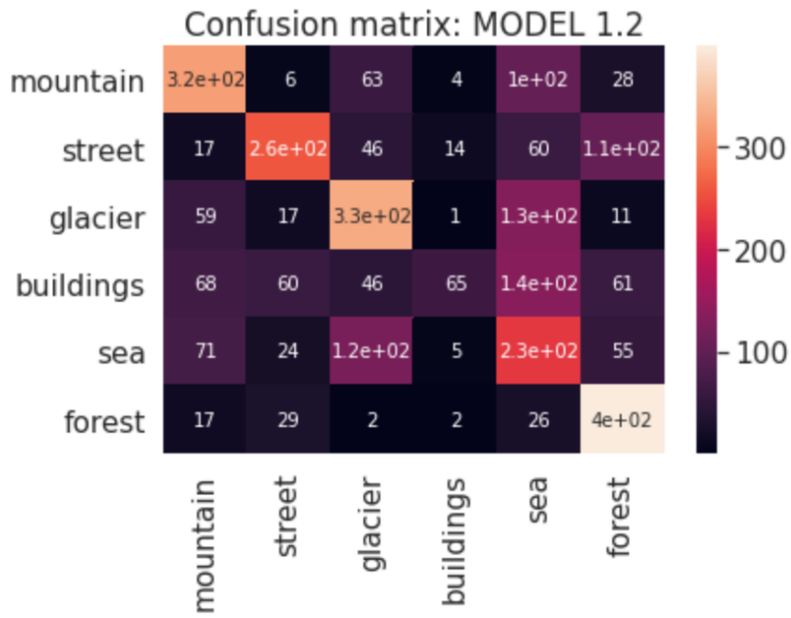

Fig. 5: Confusion matrix of model 1.2

	Mountain	Street	Glacier	Buildings	Sea	Forest
F1-score	0.59	0.58	0.57	0.25	0.39	0.7
Recall	0.60	0.51	0.60	0.15	0.46	0.84

Table 4: Recall of model 1.2

- **Model 2**
  In model 2, all the parameters are the same as model 1, but we have decreased the learning rate from 0.1 to 0.001 with the same optimization method. Comparing to model 1 the accuracy of the test and training has decreased which is because of lower learning rate.

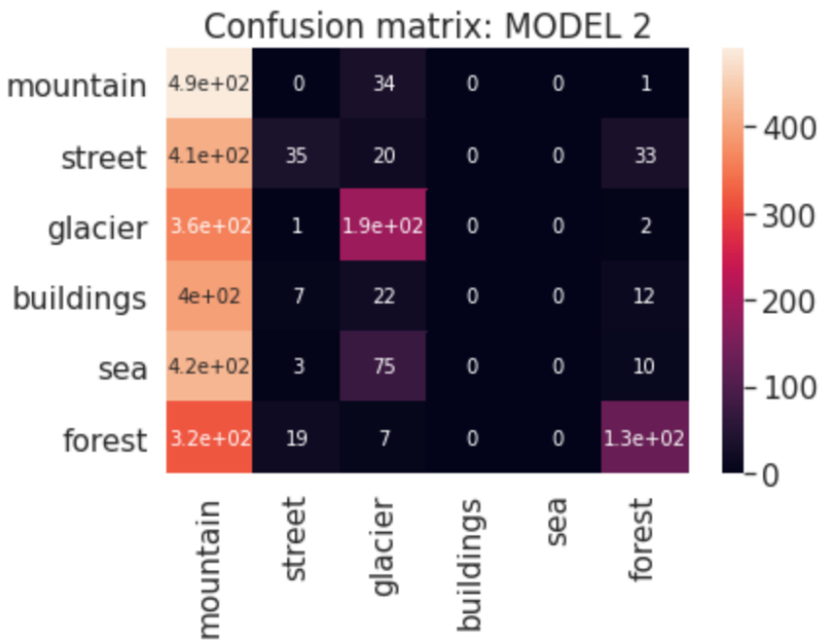

Fig. 6: Confusion matrix of model 2

	Mountain	Street	Glacier	Buildings	Sea	Forest
F1-score	0.33	0.12	0.42	0	0	0.39
Recall	0.94	0.07	0.34	0	0	0.28

Table 5: Recall of model 2

- **Model 2.1**
  In this model, we do not change the learning parameter but instead of applying SGD we will use Adam optimizer, which increases test and training accuracy to 57.56% and 68.7%, respectively.

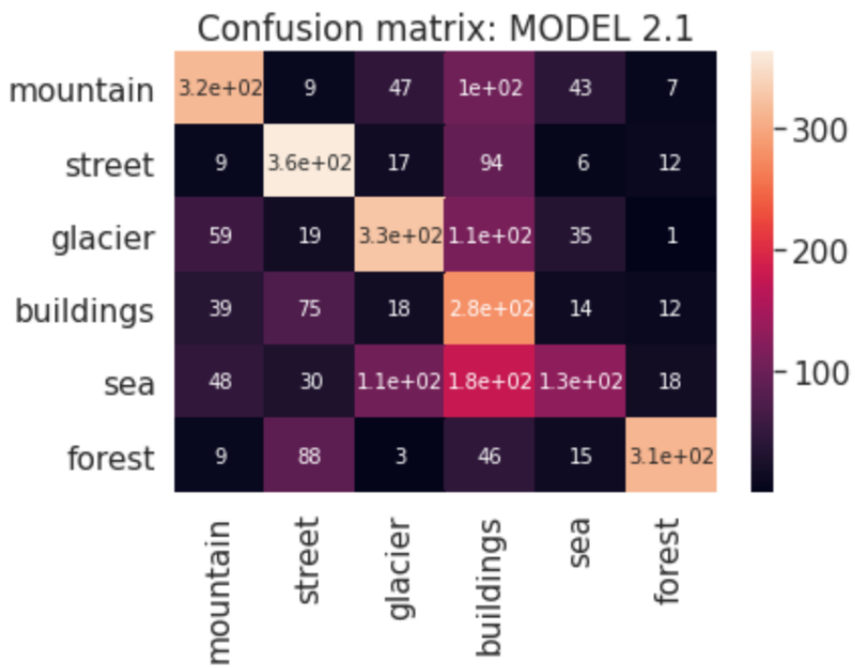

Fig. 7: Confusion matrix of model 2.1

	Mountain	Street	Glacier	Buildings	Sea	Forest
F1-score	0.63	0.67	0.61	0.45	0.34	0.75
Recall	0.6	0.72	0.59	0.64	0.25	0.66

Table 6: Recall of model 2.1

- **Model 3**
  Another parameter than can affect the model accuracy is proportion of validation split. In previous models we set it to 0.1, but in model 3 (with 3 hidden layers) it has increased to 0.2 and the new validation set increases the test error to 58.93% which shows less overfitting.

  **Considering the loss and accuracy figure-which are shown in the code- the validation does not follow the training, so we suggest increasing validation size.*

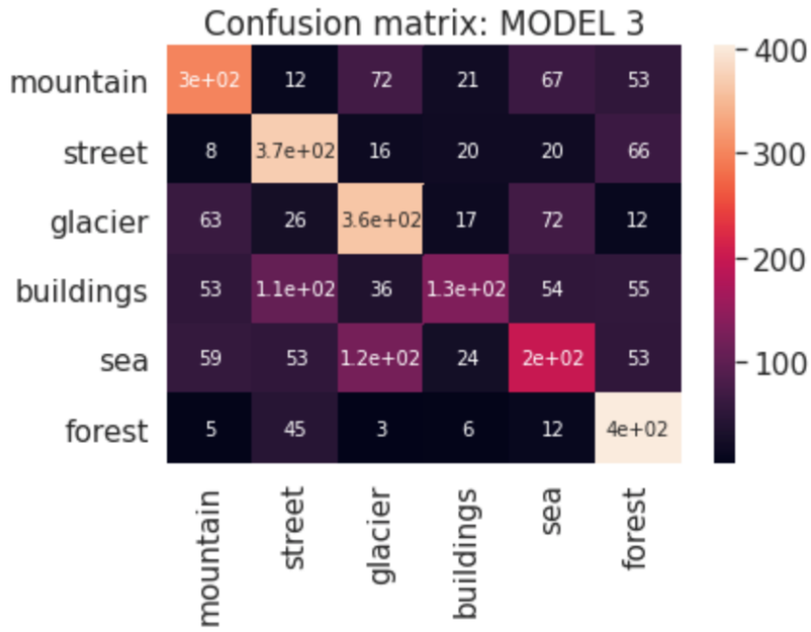

Fig. 8: Confusion matrix of model 3

	Mountain	Street	Glacier	Buildings	Sea	Forest
F1-score	0.59	0.66	0.62	0.40	0.43	0.72
Recall	0.57	0.74	0.65	0.30	0.39	0.85

Table 7: Recall of model 3

- **Model 4**

  Activation functions are one of the most important factors in a neural network which are chosen based on the type of the problem, for example linear or non-linear. In model 4 we have changed activation function of last hidden layer from ReLU to tangent hyperbolic which is suggested for multi-layer neural networks but in this case, it does not improve the accuracy of the model.

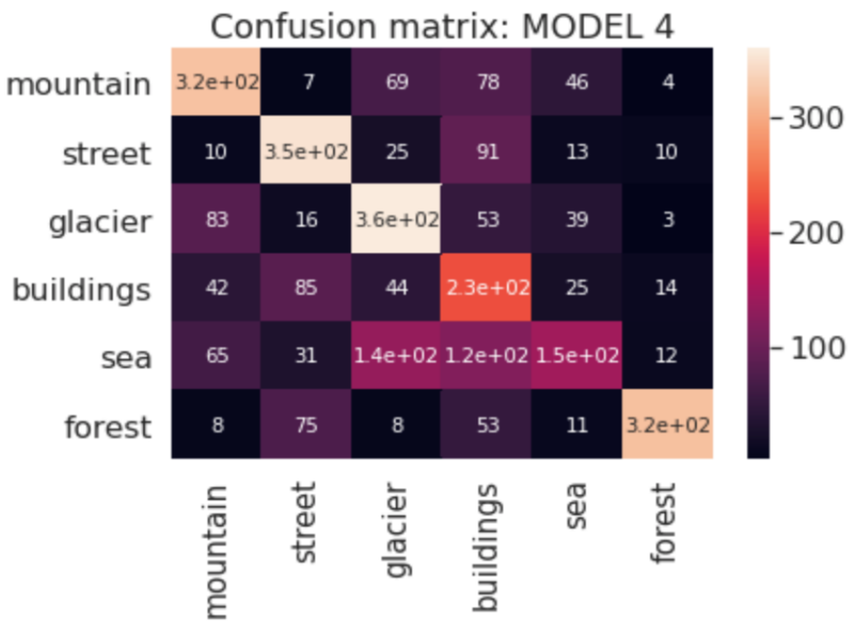

Fig. 9: Confusion matrix of model 4

	Mountain	Street	Glacier	Buildings	Sea	Forest
F1-score	0.61	0.66	0.6	0.43	0.4	0.76
Recall	0.61	0.70	0.65	0.52	0.28	0.67

Table 8: Recall of model 4

- **Model 5**
  In model 5, we have increased number of epochs from 10 to 15, so we had 15 complete passes through the training set, and it improved the training and test accuracy to 72.86% and 60.93%, respectively, which are the best gotten accuracies in implemented models.

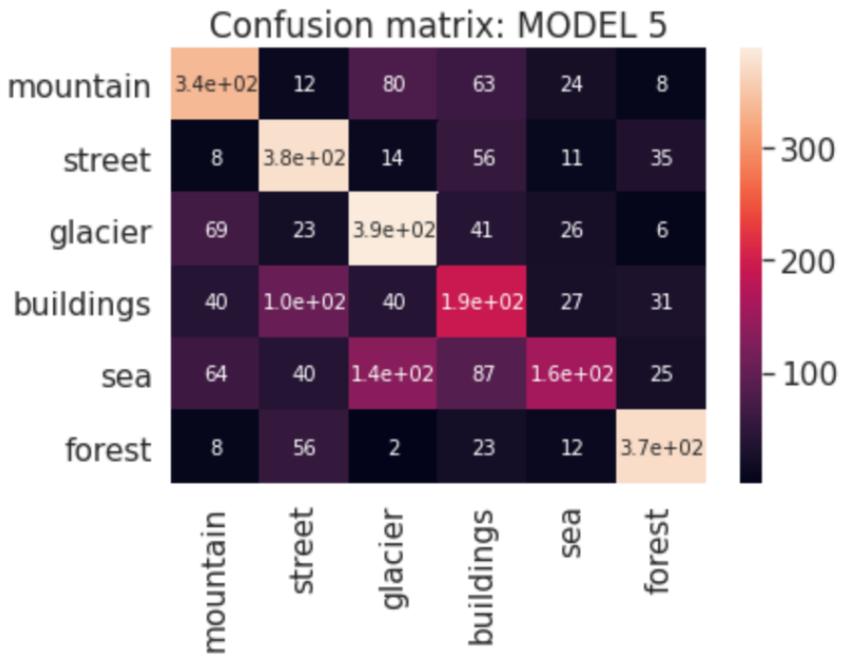

Fig. 10: Confusion matrix of model 5

	Mountain	Street	Glacier	Buildings	Sea	Forest
F1-score	0.64	0.68	0.64	0.43	0.41	0.78
Recall	0.64	0.75	0.70	0.44	0.30	0.79

Table 9: Recall of model 5

- **Model 6**
  Since we have got the best training and test accuracy, our goal is to decrease the overfitting. One of the methods to decrease the overfitting is L1 regularization which is also known as Lasso regression. By applying L1 regularization the accuracy of the model decreases dramatically.

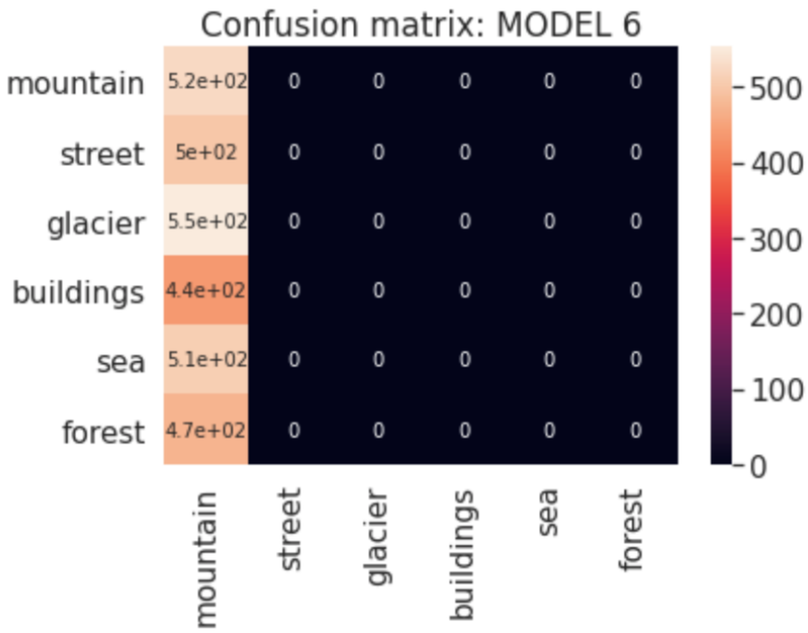

Fig. 11: Confusion matrix of model 6

	Mountain	Street	Glacier	Buildings	Sea	Forest
F1-score	0.3	0	0	0	0	0
Recall	1	0	0	0	0	0

Table 10: Recall of model 6

- **Model 6.1**
  Ridge regression or L2 regularization is another method to decrease overfitting and improve the model efficiency. In model 6.1, there is no overfitting, however the accuracy of the model is extremely low.

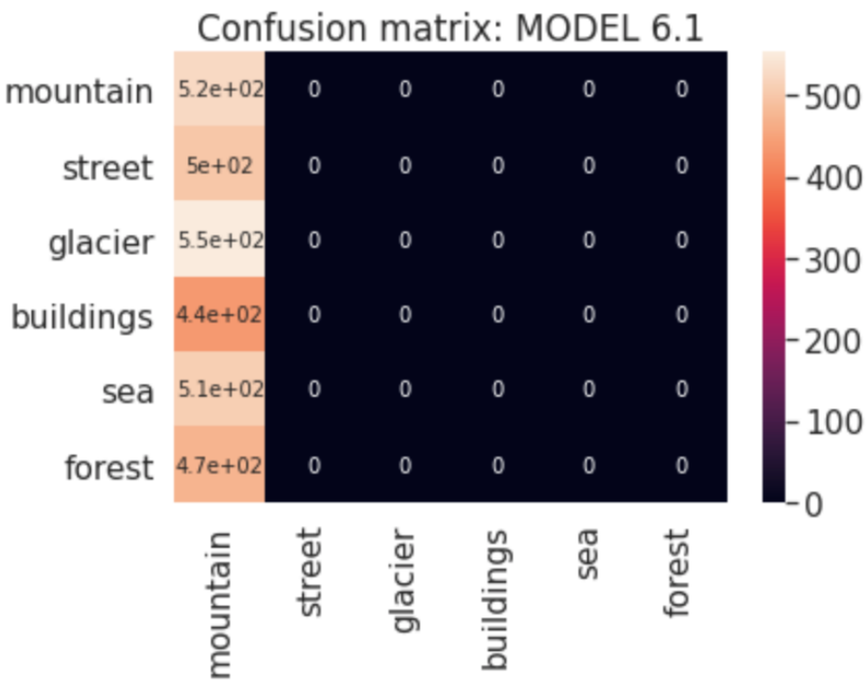

Fig. 12: Confusion matrix of model 6.1

	Mountain	Street	Glacier	Buildings	Sea	Forest
F1-score	0.29	0	0	0	0	0
Recall	1	0	0	0	0	0

Table 11: Recall of model 6.1

- **Model 6.2**
  Model 6.2 uses L1 and L2 regularization in two hidden layers which shows a reduction in training and test accuracy.

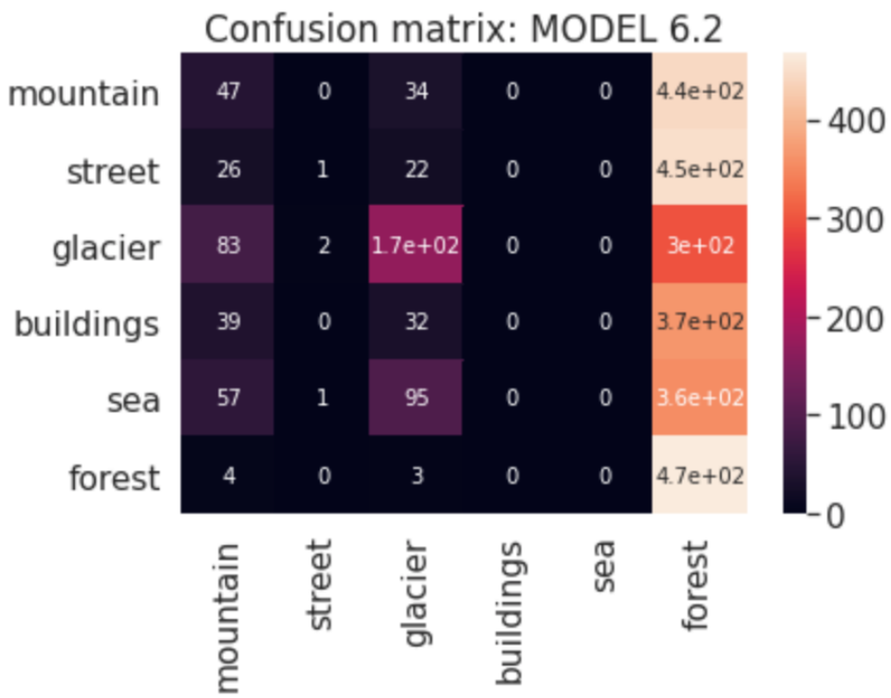

Fig. 13: Confusion matrix of model 6.2

	Mountain	Street	Glacier	Buildings	Sea	Forest
F1-score	0.12	0.003	0.37	0	0	0.33
Recall	0.09	0.01	0.31	0	0	0.98

Table 12: Recall of model 6.2

- **Model 6.3**
  Dropout is a common regularization method for neural networks to prevent overfitting. It sets the outgoing edges of neurons in hidden layers to 0 randomly. There is a dropout rate which as between 0 and 1, and in model 6.3 it is set to 0.5. Comparing to the other regularization methods, dropout works more efficiently on this model and the test accuracy is 58.17% which is one of the highest gotten accuracies.

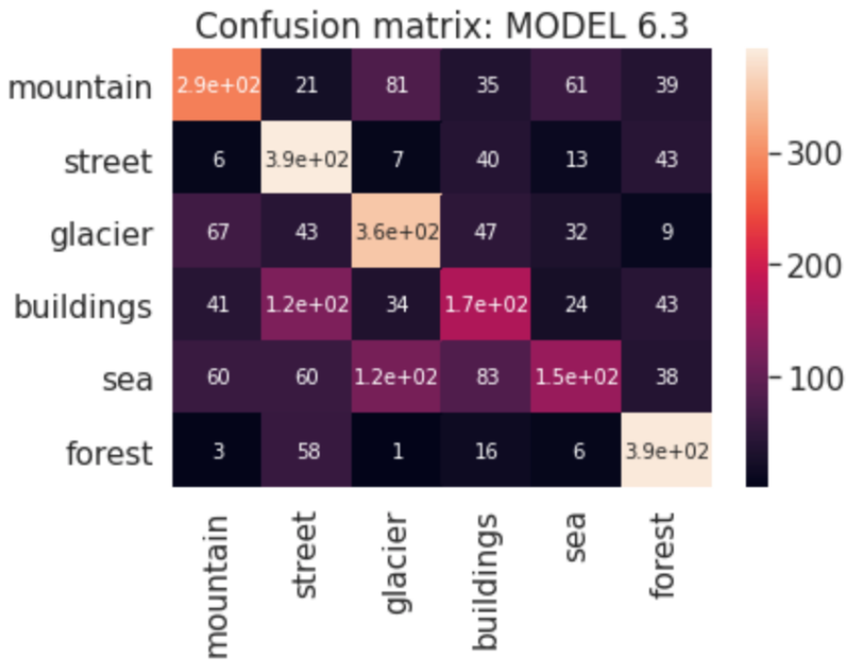

Fig. 14: Confusion matrix of model 6.3

	Mountain	Street	Glacier	Buildings	Sea	Forest
F1-score	0.58	0.65	0.62	0.41	0.38	0.75
Recall	0.55	0.78	0.64	0.39	0.29	0.82

Table 13: Recall of model 6.3

4. CNN Models

Convolutional neural network (CNN) [4] is exclusively designed to train for image classification-based problems since it follows weight sharing mechanism thus it can operate on large input data while easily extracting features using various filters and therefore it consumes small memory. In the previous part, we studied the parameters of simple neural network, so we are going to focus on the layers and the main architecture more complex neural networks.
There are three major parts in CNN namely convolution layer, pooling layer and fully connected layer, we will share a quick overview of these layers in following section, refer to fig. 15 for viewing the overall architecture of the CNN model.

- **Convolution layer** applies convolution operation on the input layer, while using a dot product operation between the input data and weights, the result is passed on to following layer, this operation changes the dimension of the resultant data since convolution filter matrix might vary.
- **Pooling layer** performs down sampling operation along the width of the input data in order to reduce spatial dimensions. In this project I have used max pooling since it takes the largest element from the given window.

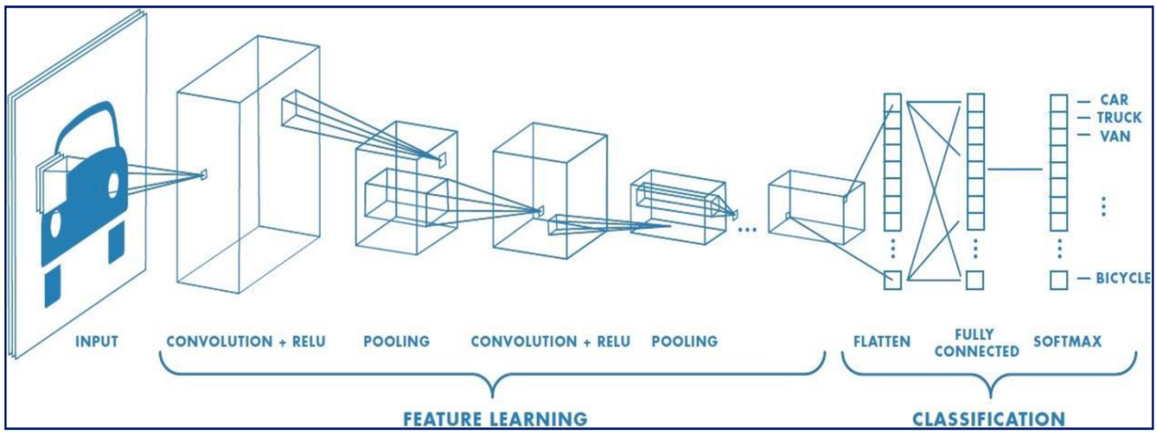

Fig. 15: Overall architecture of a CNN based model.

- **A fully connected layer** is also known as a dense layer which is typically a fully connected layer of neurons which receive inputs from previous layers, have their own weights and bias. Output of a fully connected layer depends upon the output of activation function used in that layer therefore activation refers to activation function, kernel refers to weights associated with that neuron, refer to fig. 15 which shows overall architecture of CNN models.

In the previous part, we studied the parameters of simple neural network, so we are going to focus on the layers and the main architecture of the more complex neural networks.

It is evident from fig.16 that there are 6 convolution layers used in this model, each with ReLU as an activation function. Another activation function option is Leaky ReLU and the major difference between **leaky ReLU** and ReLU is the ability of leaky to obtain negative values while latter cannot attain it.

ReLU activation function returns element-wise max(x, 0) for all x>0, else it is 0, this provides very simple and fast computation abilities in comparison to sigmoid or tanh functions. Output is always linear but non-negative, therefore it generalizes input data by setting negative inputs to 0 during final output unlike sigmoid function.

Padding is kept same so that bits are padded evenly from both left and right side and uneven padding can be avoided.

Whereas **Max Pooling** size is fixed as 2x2, the pooling layer helps in reducing the dimension of the incoming data values by selecting the maximum values from the 2x2 pool.

Dropout Regularization is set to 0.5, this helps in dropping out random values while training, this implies that contribution of activation functions by dropped out neurons is ignored in the forward pass, while their weights are not updated during the backward pass.

The final layer is a **flattened dense layer** as described above (Fig. 15) which uses Softmax as activation function in the output layer because it gives probabilities as output for classifying given inputs, therefore sum of this function for a given input is always one and output can be easily determined by selecting the one having maximum probability. Using these parameters CNN is compiled successfully and in upcoming section we will discuss the results in detail.

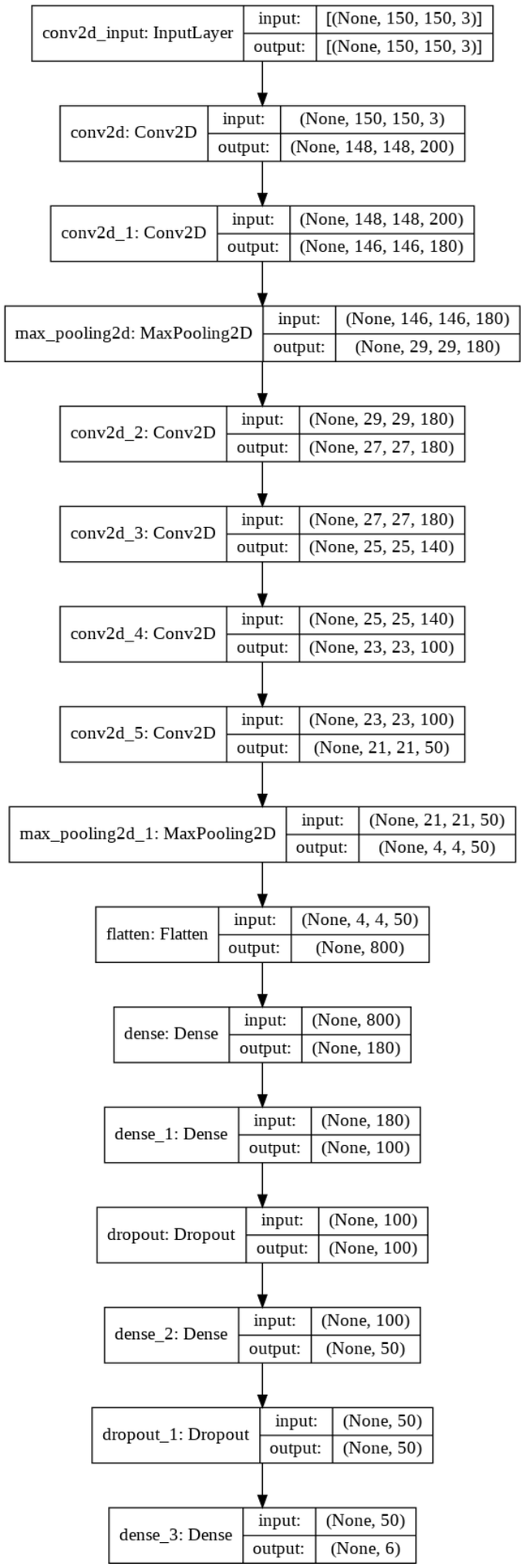

Fig. 16: Architecture of the best CNN model (highest accuracy)

In this project we have implemented three CNN model, one as the baseline model and improvements of the first model. Using the result of MLP models, we knew Adam optimizer and Dropout regularization methods perform accurately on this dataset, so we tried the same methods to see whether they are helpful in CNN or not. Table 14 shows that the CNN model was successful and both training ad test accuracy have raised. In the second model, we have decreased the learning rate and size of training set in cross validation and the test accuracy has increased by 2.2%, but we are still facing overfitting by 13.3%. To tackle the overfitting issue, we have added another Dropout regularization layer with 0.5 rate, and as you can see the test accuracy has increased to 86.1% and also overfitting percentage has decreased.

	Learning rate	# Epochs	Reg.	Val. Set	Train Accuracy	Test Accuracy
1	0.001	35	Dropout	0.2	94.56%	82.1%
2	0.0001	35	Dropout	0.3	97.6%	84.3%
3	0.0001	40	Dropout2	0.3	97.5%	86.1%

Table 14: Parameters and accuracy of CNN models

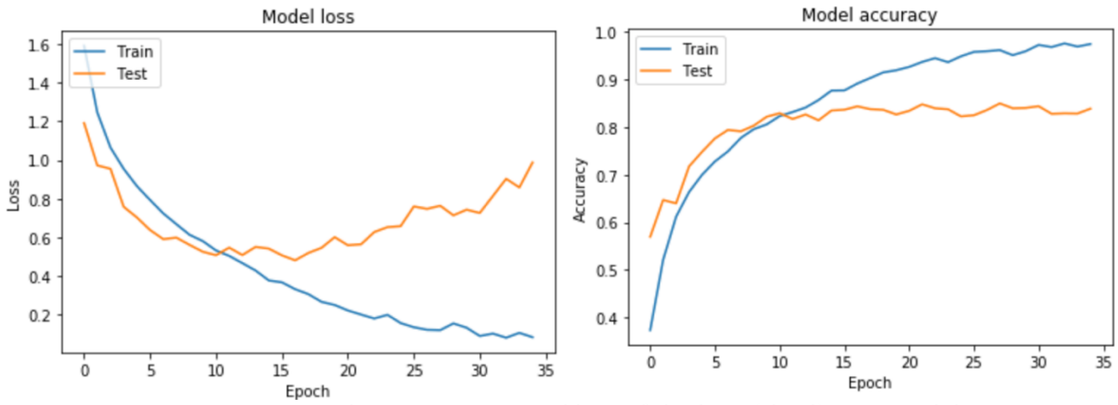

Fig. 17: Test and train accuracy and loss of the best (third) CNN model

The results show the advantage of using CNN versus MLP models in this problem.

5. ResNet-50

After AlexNet [5] popularized convolutional neural networks for image classification, there have been several modifications and improvements to the network architecture. New convolutional architectures have provided improvements to the ImageNet image classification task every year since 2012. Various models have explored different aspects of building neural network architectures. One common theme among them being increase in the depth of the models. After AlexNet won ILSVRC-2012, VGGNet [6] was a 16 layered deep convolutional neural network. It showed that the depth of the network is a critical factor in performance. VGGNet was the runner up of ILSVRC-2014. That year the competition was won by GoogLeNet [7], a 22 layer model. The main contribution of the GoogLeNet model was the Inception Module that reduced the number of

parameters of the model drastically. The inception module consists of 1x1, 3x3, and 5x5 convolutions. Additionally, it contains a parallel pooling layer. Figure 2.3 shows the inception module used in GoogLeNet. The GoogLeNet model has 9 inception modules. It replaced the fully connected layer, which contributes the most parameters, with a global average pooling layer. GoogLeNet has only 5 million parameters as compared to AlexNet's 60 million. Microsoft Research Asia's ResNet model was the winner of ILSVRC-2015. An important observation made in the ResNet paper by He et al. [8] is that simply stacking additional convolutional layers does not give a better model. They compared a 56 layer convolutional network with a 20 layer network. The experiment showed that the deeper 56 layered network had a higher training and test error on the CIFAR-10 dataset. They also observed a similar phenomenon with the ImageNet dataset.

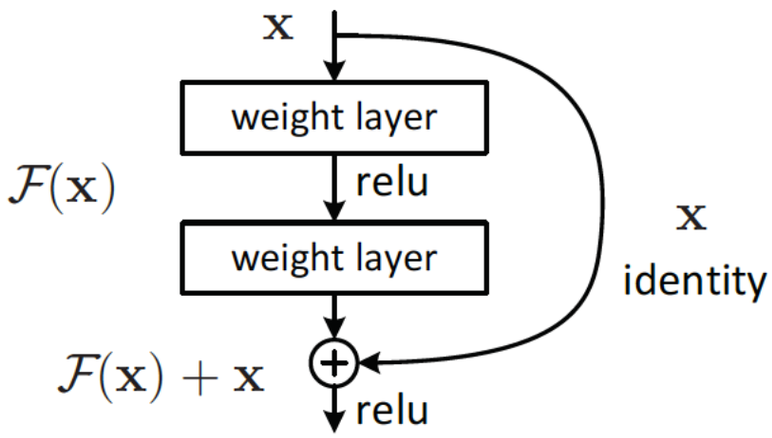

Fig. 18: Residual block from the ResNet architecture [8]

The primary contribution of the ResNet model is the residual block. The residual block addresses the problem of vanishing gradients while training very deep models by introducing skip connections with identity mapping. These are skip connections with fixed weight of 1.0. The skip connections allow the model to grow very deep and still be trainable. Figure 18 shows the basic residual block [6]. Consider this network without the skip connections. We expect it to learn some underlying mapping H(x) by learning the 2 weight layers. The residual block instead tries to fit to another mapping F(x) such that,

$$H(x) = F(x) + x$$

Here F(x) is a residual mapping with respect to identity. If the optimal mapping is identity then it is easy to learn the weights as 0. Otherwise, if the optimal mapping is close to identity, it is easier to find small fluctuations in addition to the identity mapping.

Considering the high complexity of ResNet and the results of the CNN models, we anticipate high training accuracy while the test accuracy must be low. Same as previous models, we have applied Adam optimizer with $1x10^{-4}$ and validation split is set to 0.3. As anticipated, the training accuracy was ***99.46%*** while the test

accuracy was ***57.13%*** with only 5 epochs, which shows high overfitting and proves the idea of high complexity of ResNet for this problem.

6. Results and Conclusions

The goal of this project is to study the parameters and different model architectures for a multi-class image classification problem. We started with simple MLP models and in each model changed a parameter to observe the effects of that. The next step was trying to improve the performance of the utilized models. As shown in table 15, CNN model had the best performance for the classification problem and increasing the complexity by ResNet caused overfitting.

	MLP	CNN	ResNet-50
Training Accuracy	72.86%	97.5%	99.46%
Test Accuracy	60.93%	86.1%	57.13%

Table 15: The test and training accuracy of best models in each type

Finally, after comparing different implemented models in previous sections we have observed:

- Increasing number of hidden layers can increase the efficiency of the model, but we must also prevent overfitting.
- Decreasing and increasing learning rate may decrease the accuracy of the model, so it is suggested to find the best one.
- Adam optimizer works better than SGD in this study.
- If the validation accuracy does not follow training accuracy, you can increase validation split proportion.
- Increasing number of epochs will increase the training accuracy.
- Dropout is more suitable than L1 and L2 regularization in this neural network.
- If during training a model, the growth of training and test accuracy is stopped, using a more complex model can help.
- CNN models are more suggested for vision tasks, compared to MLP models
- If the training accuracy is high, using a more complex method does not improve the result, but causes high overfitting.

For further study, we suggest data augmentation to increase the training data size. Unfortunately, because of lack of RAM and GPU we could not try more models and data augmentation.

Bibliography

[1] P. Bansal, "Kaggle," [Online]. Available: https://www.kaggle.com/puneet6060/intel-image-classification.

[2] Jianglin Huang, Yan-Fu Li, Min Xie, "An empirical analysis of data preprocessing for machine learning-based software cost estimation," *Information and Software Technology,* vol. 67, no. 0950-5849, pp. 108-127, 2015.

[3] J. Tang, C. Deng and G. Huang, "Extreme Learning Machine for Multilayer Perceptron," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 27, pp. 809-821, 2016.

[4] O'Shea K., and Nash R., "An introduction to convolutional neural networks," *arXiv,* 2015.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems, NIPS,* vol. 12, p. 1097–1105, 2012.

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for largescale image recognition," in *International Conference on Learning Representations*, 2015.

[7] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR),*, 2016.

My biography:

I have studied my B.Sc. of Elcetrical Engineering at IKIU in Iran (2014-2018). I moved to California to continue my studies in summer 2019. I am going to be graduated this semester and also, I am going to continue my education as a Ph.D. student at Cornell Tech. My interest is in optimization and modelling under uncertainty.

The title of my thesis is "*Generative learning for cancer detection using hyperspectral images"*. I have used sparse optimization methods to generate new data for classification problem. Also, you are in my committee, I hope you like my thesis.

Thank you very much

Rojin Zandi