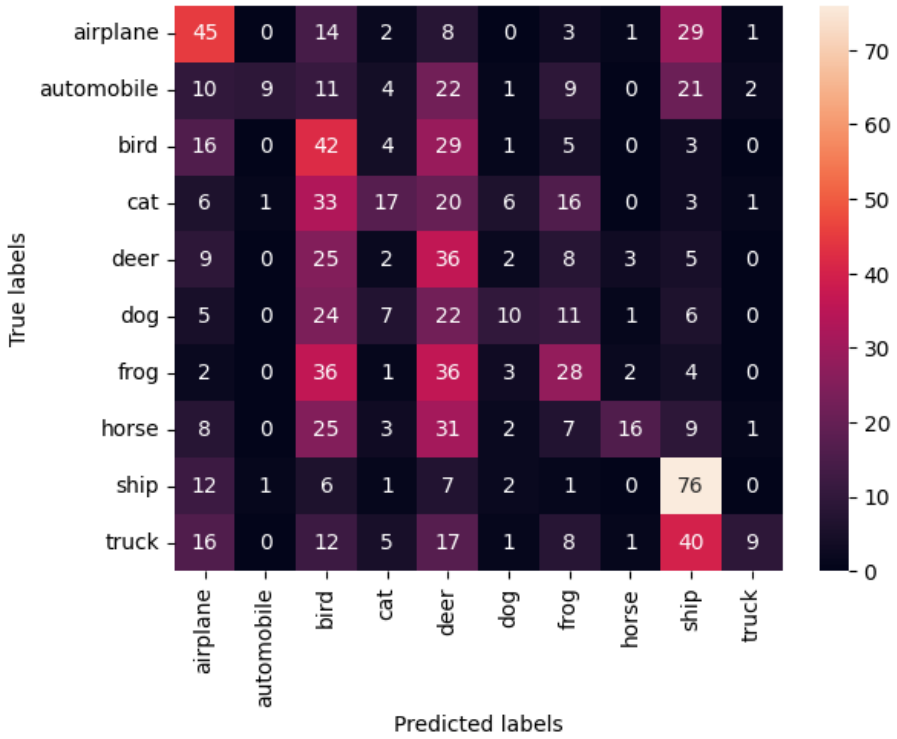
Computer Vision Homework 1

Oron Barazani 305295818 & Roee Hadar 206390098

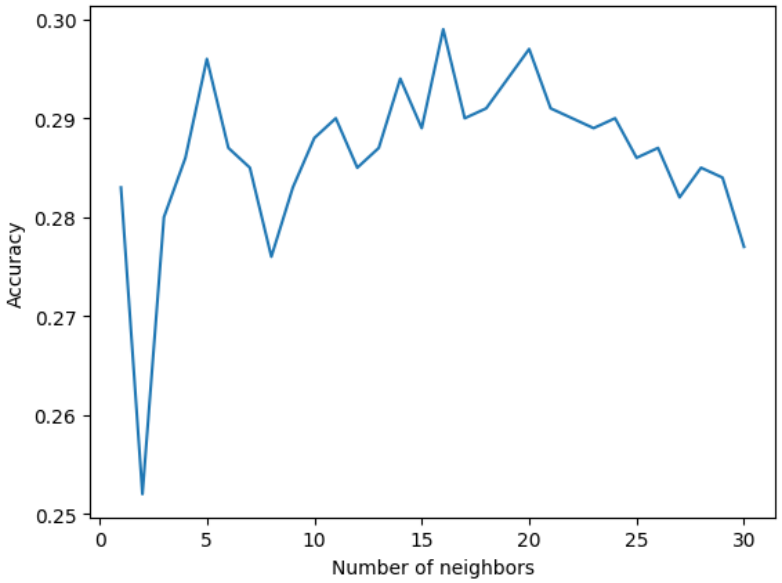
Part 1:

1. Given below are 5 pictures from the CIFAR10 dataset:

2. The results we achieved in this section on the test set is $accuracy = 0.39$. The accuracy is fairly low – kNN as we know is not a good model for classifying images – it doesn't handle high dimensional vectors well, and it is very sensitive to noise.
3. After loading 1000 test data from the CIFAR10 model we achieved $accuracy = 0.29$ on the test set. Moreover, given below is a table of the predicted label against the ground truth label. It is noticeable that similar object in images, like ship and trucks, which share similar metal elongated body, are frequently mixed by the model.prediction.



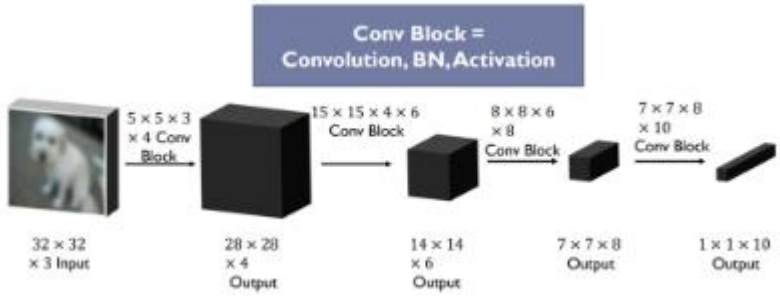
4. Provided below is a graph of the K Nearest Neighbor algorithm against the Number of neighbors chosen:



As we can see from the graph, when using a small number of neighbors we achieve poor results. As we increase the number of neighbors, we get better results until a certain threshold, where the results start to plummet in accuracy. This phenomenon is a classic example for the tradeoff between Bias and Variance - when we take a small number of neighbors, we have large variance, hence the predictions are poor, and when we take a high number of neighbors the model achieves a large bias on the training set, hence achieving once again lower accuracies.

Part 2:

1. In this section, we have used the CNN architecture as shown in the Tutorials:



The Hyper-parameters we used were

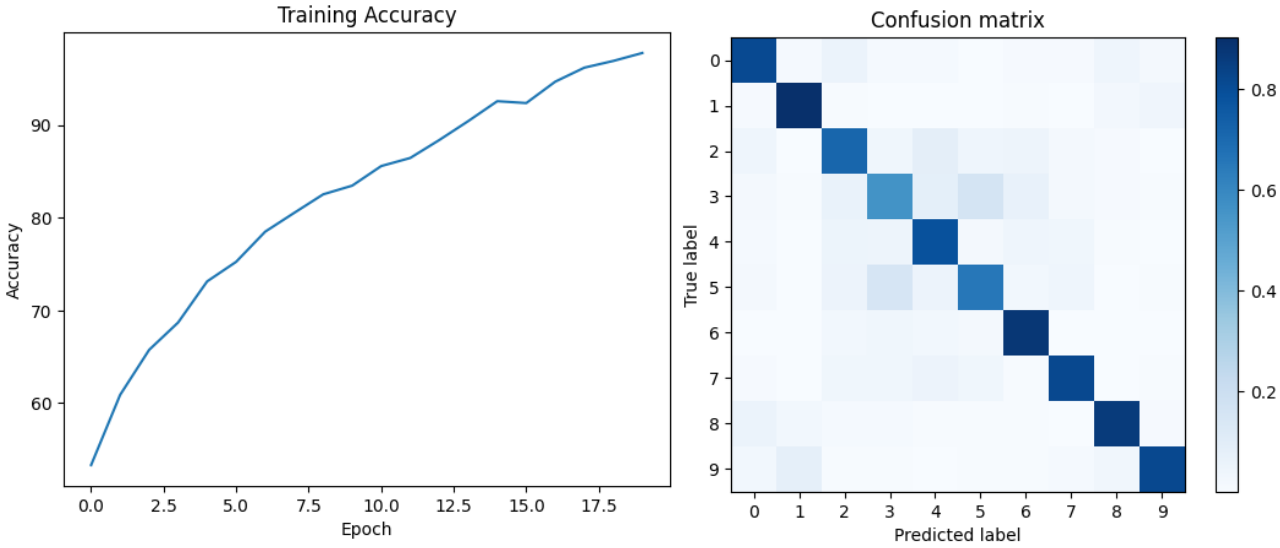
$$Batch\ Size = 128, \qquad Learning\ Rate = e^{-4}, \qquad Epochs = 20$$

The data set used is SVHN (Street View House Numbers)

The model has been trained locally on RTX2080 GPU in $114.57\ [s]$

With training accuracy 97.76% on the last epoch, and accuracy of $\mathbf{78.06\%}$ **on the test set**

Provided below are the Accuracy graph and the Confusion matrix:



As can be seen from the Confusion matrix, labels 3 and 5 are the least certain when compared to the other labels (which means that dog and cat are similar for the algorithm). Labels 2, 7, 8, 9 predict their true label much more precisely.

2. We planned **ResNet** with the help of GPT model. The model is composed of conv-batchNorm-PreLU and immediate after, 4 varying sizes layers, each of them composed of 2 sequential layers of **residual blocks**, and at the end, fully connected layer. Each **residual block** is composed of two conv-batchNorm layers with PreLU between them, but there is a shortcut that let data flow "skipping" those layers. The ResNet types of models let gradients flow network in the path which is optimal. It has been shown that it helps reducing the vanishing gradient effect. We also introduced **PreLU** which also help

to deal with vanishing gradients, as it doesn't "kill" negative values. It also more flexible as it contains learnable hyper-parameter for the slope of the function, which can be adjusted per data.

The input dimensions for the model are $W = 32,\ H = 32,\ C = 3$ image of CIFAR10 dataset. The output of the model is 10-dimensional vector, represent probabilities for each class of 10 classes of the dataset.

The summary of the ResNet model:

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 32, 32]	1,728
BatchNorm2d-2	[-1, 64, 32, 32]	128
PReLU-3	[-1, 64, 32, 32]	1
Conv2d-4	[-1, 64, 32, 32]	36,864
BatchNorm2d-5	[-1, 64, 32, 32]	128
PReLU-6	[-1, 64, 32, 32]	1
Conv2d-7	[-1, 64, 32, 32]	36,864
BatchNorm2d-8	[-1, 64, 32, 32]	128
PReLU-9	[-1, 64, 32, 32]	1
ResidualBlock-10	[-1, 64, 32, 32]	0
Conv2d-11	[-1, 64, 32, 32]	36,864
BatchNorm2d-12	[-1, 64, 32, 32]	128
PReLU-13	[-1, 64, 32, 32]	1
Conv2d-14	[-1, 64, 32, 32]	36,864
BatchNorm2d-15	[-1, 64, 32, 32]	128
PReLU-16	[-1, 64, 32, 32]	1
ResidualBlock-17	[-1, 64, 32, 32]	0
Conv2d-18	[-1, 128, 16, 16]	73,728
BatchNorm2d-19	[-1, 128, 16, 16]	256
PReLU-20	[-1, 128, 16, 16]	1
Conv2d-21	[-1, 128, 16, 16]	147,456
BatchNorm2d-22	[-1, 128, 16, 16]	256
Conv2d-23	[-1, 128, 16, 16]	8,192
BatchNorm2d-24	[-1, 128, 16, 16]	256
PReLU-25	[-1, 128, 16, 16]	1
ResidualBlock-26	[-1, 128, 16, 16]	0
Conv2d-27	[-1, 128, 16, 16]	147,456
BatchNorm2d-28	[-1, 128, 16, 16]	256
PReLU-29	[-1, 128, 16, 16]	1
Conv2d-30	[-1, 128, 16, 16]	147,456
BatchNorm2d-31	[-1, 128, 16, 16]	256
PReLU-32	[-1, 128, 16, 16]	1
ResidualBlock-33	[-1, 128, 16, 16]	0
Conv2d-34	[-1, 256, 8, 8]	294,912
BatchNorm2d-35	[-1, 256, 8, 8]	512
PReLU-36	[-1, 256, 8, 8]	1
Conv2d-37	[-1, 256, 8, 8]	589,824
BatchNorm2d-38	[-1, 256, 8, 8]	512
Conv2d-39	[-1, 256, 8, 8]	32,768
BatchNorm2d-40	[-1, 256, 8, 8]	512
PReLU-41	[-1, 256, 8, 8]	1
ResidualBlock-42	[-1, 256, 8, 8]	0
Conv2d-43	[-1, 256, 8, 8]	589,824
BatchNorm2d-44	[-1, 256, 8, 8]	512
PReLU-45	[-1, 256, 8, 8]	1
Conv2d-46	[-1, 256, 8, 8]	589,824
BatchNorm2d-47	[-1, 256, 8, 8]	512
PReLU-48	[-1, 256, 8, 8]	1
ResidualBlock-49	[-1, 256, 8, 8]	0
Conv2d-50	[-1, 512, 4, 4]	1,179,648
BatchNorm2d-51	[-1, 512, 4, 4]	1,024
PReLU-52	[-1, 512, 4, 4]	1
Conv2d-53	[-1, 512, 4, 4]	2,359,296
BatchNorm2d-54	[-1, 512, 4, 4]	1,024
Conv2d-55	[-1, 512, 4, 4]	131,072
BatchNorm2d-56	[-1, 512, 4, 4]	1,024
PReLU-57	[-1, 512, 4, 4]	1

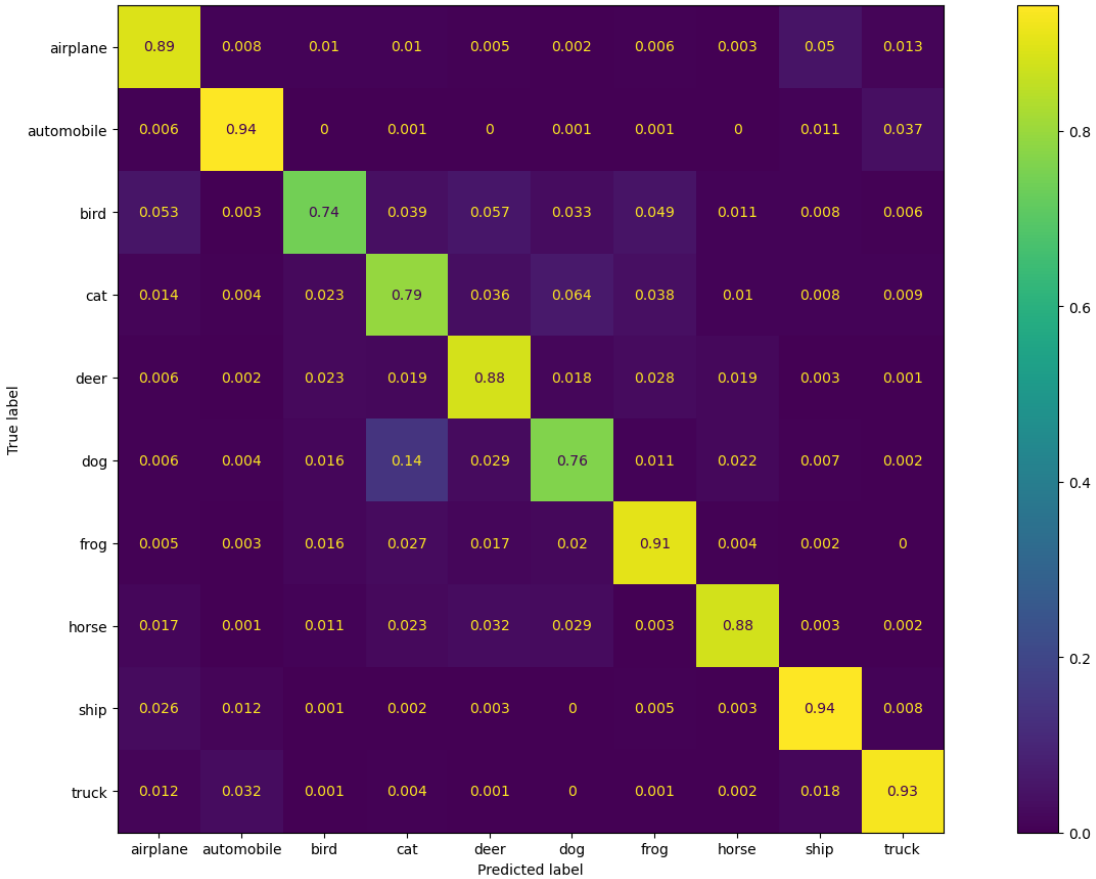
```
    ResidualBlock-58            [-1, 512, 4, 4]               0
          Conv2d-59            [-1, 512, 4, 4]       2,359,296
     BatchNorm2d-60            [-1, 512, 4, 4]           1,024
            PReLU-61            [-1, 512, 4, 4]               1
          Conv2d-62            [-1, 512, 4, 4]       2,359,296
     BatchNorm2d-63            [-1, 512, 4, 4]           1,024
            PReLU-64            [-1, 512, 4, 4]               1
    ResidualBlock-65            [-1, 512, 4, 4]               0
AdaptiveAvgPool2d-66            [-1, 512, 1, 1]               0
          Linear-67                  [-1, 10]           5,130
================================================================
Total params: 11,173,979
Trainable params: 11,173,979
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 15.50
Params size (MB): 42.63
Estimated Total Size (MB): 58.14
```

---

3. To train the model we split the train data into train and validation set. This allows us to estimate model performance over unseen data while training. The model has been trained for number of epochs (20) and took best model in terms of validation accuracy. This set the "number of epochs" hyper-parameter to 18. The next hyper-parameter, batch size, was compromise between GPU capabilities and training speed. It was set to 32.

The results:

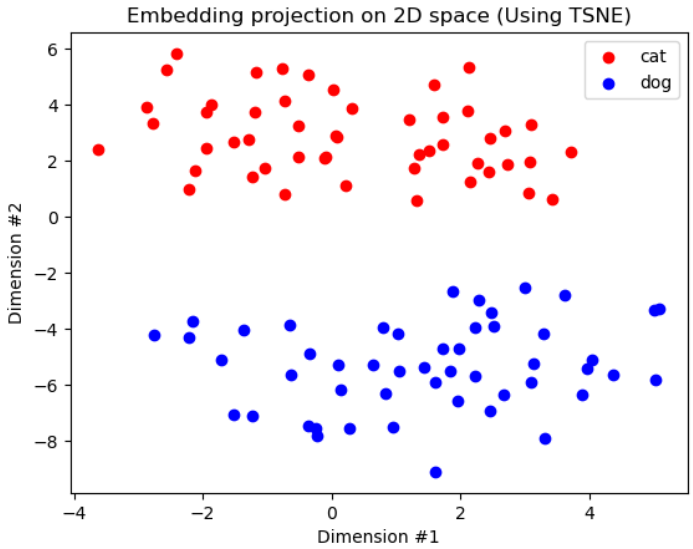
```
Best Validation Accuracy: 85.89%
Test Accuracy: 86.69%
```



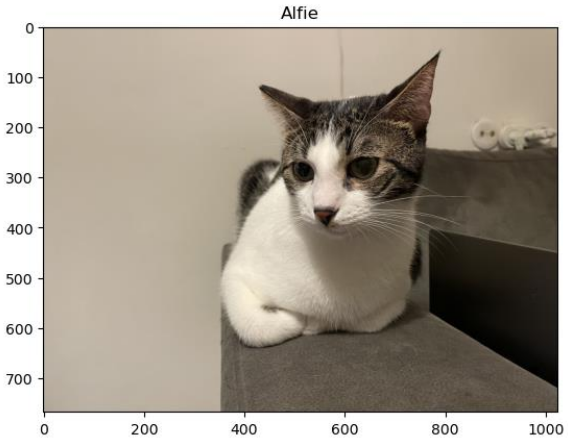
As we can see, the performance is much better in terms of accuracy compared to previous classifier. It also worth mentioning that the hard classes to classify for the previous classifier (dog, cat, bird) are also difficult to this classifier also. Most of the wrong classifications are dog mistaken for cats and vice versa.

Part 3:

1. Using t-SNE algorithm we projected the 768th dimensional vector to the 2D space. t-SNE tries to minimize the Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding, meaning that drawn from the same distribution in the high dimensional space will be likely close in the 2D space. We can see that the distribution of dogs images and cats images are indeed form clusters in the 2D space, suggesting that they were close in means of distribution in the embedding space.



2. We loaded Alfie's image and used CLIP to encode it to the embedding space. in the embedding space we used cosine similarity measure to find its nearest data points.

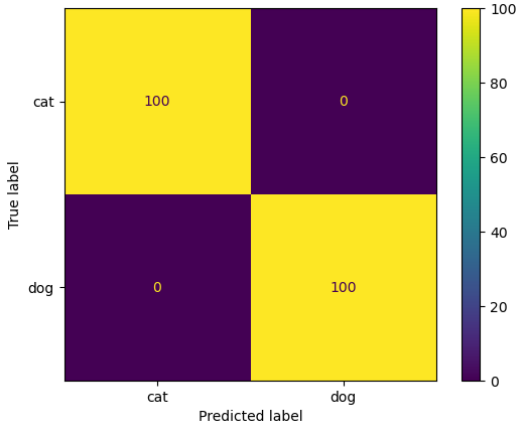


The most similar cat is making sense, same colors and pattern of fur. The second cat is similar by fur pattern although he is ginger. The last 3 kind of resemble Alfie by their color (gray with black strips), hence their lower similarity score.

3. We took all the test images and encoded them to embedding space. next, we encode the following sentences:

```
"a photo of a cat", "a photo of a dog"
```

We used cosine similarity to check each image similarity to each sentence. If an image had higher cosine similarity with the first sentence, it was classified as a cat, else it was classified as a dog. We check the overall performance, in means of accuracy and confusion matrix and got perfect result.



The strength of using CLIP for classification application is that the model doesn't need label images for training. CLIP model "understand" image by it semantic content. The down side of this approach is that we get score not in mean of probability, and we "have to" classify each image to one of the labels of cat or dog, even though it can be a photo of neither of them.
For other class classification, it might be needed to add another label – maybe "a photo of an animal", and check cosine similarity with those 3 labels. After that step, apply softmax layer to convert the score to probabilities. The probability threshold for classification is 0.5. this will add another option where every probability is lower than 0.5, and will be classified as "Other".

4. We created a function named "count_cats" which check cosine similarity of cats image with the following sentence:

```
"a photo of one cat", "a photo of two cats", "a photo of three cats", "a photo of four cats"
```

The highest measure determined the number of cats the model classified.

Part 4:

1. Using BoW algorithm for flags detection may provide both Pros and Cons. Described below is a list with explanations for Pros and Cons:
   Pros:
   - Simplicity - Bag of Words is straightforward to implement and doesn't require complex algorithms.
   - Efficiency - Once the vocabulary is built, BoW can be computationally efficient for image region representation. Moreover, the representation can be stored efficiently, especially when using sparse matrices.
   - Robustness - Considering specific flags which have unique attributes, BoW may significantly decrease the complexity of the task.

   An example for the Pros with a given Bag of Words is identification of Israel flag using the Blue Magen David on a white background as a feature, or a "word"

   Another example is attempting to identify the United States flag using as an attribute or a "word" the blue section with 50 white stars on it.

Cons:

- Ignoring Layout - Flags have specific patterns and spatial arrangements, for example stripes, crosses, stars etc., which BoW usually ignores, leading to potential misclassification,
- Pattern Recognition - BoW cannot distinguish between different flags that share similar visual elements but differ in their spatial arrangement (for example the United States flag vs. the Malaysian flag)
- Feature Discrimination - BoW might not capture the distinct features needed to distinguish between visually similar flags, especially those with minor but crucial differences.

An example for the Cons is the British flag against the Australian flag, which consists of the British flag in its top left side.

Another example is Guinea and Mali flags, which consists of Red, Yellow and Green vertical stripes in different arrangement. Such arrangement may be identified using Bag of Words only when considering spatial placement of the flag against the "words" chosen.

2. Provided below is the table of the Output Dimensions and the Number of parameters for each Layer:

Layer	Output dimensions	Number of parameters (weights)
$INPUT$	$64 \times 64 \times 3$	0
$CONV7-16$	$62 \times 62 \times 16$	2352
$POOL2$	$31 \times 31 \times 16$	0
$CONV7-32$	$29 \times 29 \times 32$	25088
$POOL2$	$14 \times 14 \times 32$	0
$FC-3$	$1 \times 3 \times 1$	18816

3.

a. Let us discuss each contributing factor with relation to overfitting:
   - Learning Rate - A very high learning rate can cause the model to converge too quickly to a suboptimal solution, which in turn makes the model learn the training data precisely, causing overfitting. If it is known that the model is overfitting the data, a likely mitigation needed is to lower the learning rate in order to achieve slower convergence, making the solution more general.
   - Batch Size - A very small batch size can cause the model to update too frequently, potentially leading to overfitting by adjusting too closely to the specific batch noise. If it is known that the model is indeed overfitting, a needed mitigation is lowering the Batch Size to let the model have more variance with respect to the data.

- Number of Layers - A deep neural network with many layers may easily overfit due to its high capacity to learn complex patterns, including added noise from the training data. A mitigation with might help with overfitting is an addition of regularization such as L1 or L2 regularization. Another option is an addition of Dropout or Batch Normalization.
- Training set size - A small test size provides less data for the model to learn from, making it more likely to memorize the data rather than generalizing from it. To avoid such instance, we may use Augmentations on the data to vary the training data and avoid memorizing it.
- Test Set Size - The test size does not affect the overfitting phenomenon, but rather checks if the phenomenon occurs.

b. The role of Dropout is as follows:
   - During Training - Dropout is a regularization technique that involves randomly "dropping out" a fraction of the neurons in a neural network layer during each training iteration with a given probability. By randomly omitting neurons during training, dropout prevents the network from becoming too reliant on specific neurons, thereby forcing it to learn more robust features that are distributed across different neurons.
   - During Testing - During testing dropout is turned off, which means that it has no effect when testing the network.
   - Reduction in overfitting - Dropout prevents overfitting by randomly omitting neurons, discouraging neurons from co-adapting to the training data. Neurons cannot rely on the presence of specific other neurons and must function independently, leading to more robust feature learning. By making the network less sensitive to specific neuron activations and spreading the learned features across a wider set of neurons, dropout reduces the likelihood of the network to overfit to the training data.