<u>Deep Learning HW 2</u>

Maxim Katz – 322406604, Yuval Levi – 325120384

<u>Assignment purpose</u>:

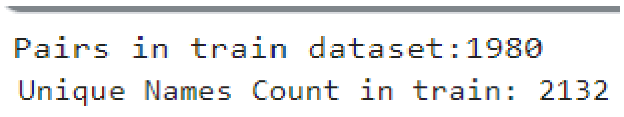
Using convolutional neural networks (CNNs) to carry out the task of facial recognition. Based on the paper Siamese Neural Networks for One-shot Image Recognition we implemented a one-shot learning task for previously unseen objects. Given two facial images of previously unseen persons, our architecture determines whether they are the same person.

<u>Data</u>:

In this assignment we used the dataset Labeled Faces in the Wild that contains labeled face images. For the one-shot facial recognition task we used this dataset to create pairs of face images and a label – 1 for match face images and 0 for pairs that are not showing the same person.

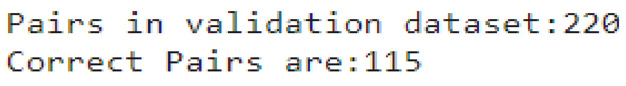
- **Data example:**

- **Pre-processing:**
  We imported the dataset Labeled Faces in the Wild from the package torchvision.datasets using the code:
  from torchvision.datasets import LFWPairs
  When loading the dataset it splits into training and testing sets.
  We split up the training set into 90% training and 10% validation.
  After the splits we have 1980 pairs in the training set when 985 pairs are labeled as correct(1) and there was 2132 different faces in the train set before splitting it into valdation.



```
Pairs in train dataset:1980
Unique Names Count in train: 2132
```

In the validation set we have 220 pairs when 115 pairs are labeled as correct(1)



```
Pairs in validation dataset:220
Correct Pairs are:115
```

In the testing set we have 1000 pairs when 500 pairs are labeled as correct (1) and there are 963 different faces.

```
Pairs in test dataset:1000
Correct Pairs are:500
```

```
Unique Names Count in test: 963
```

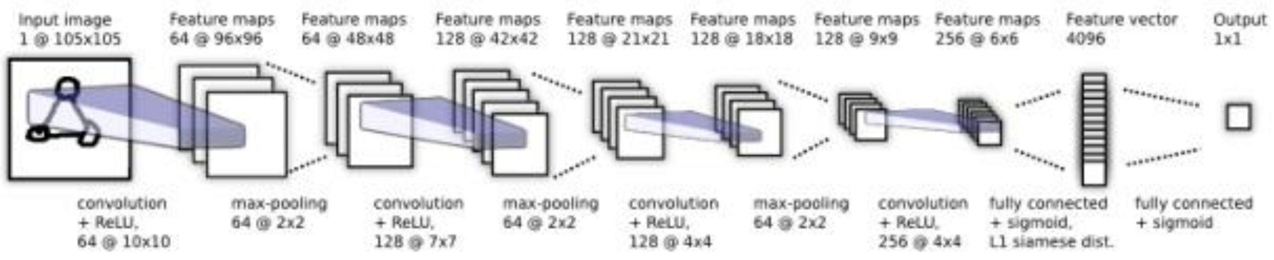
We transformed the colored images into a gray scale images to reduce dimensions, convert to tensors, and resized the images from 150x150 resolution to 105x105 resolution as seen in the article.

We created 3 DataLoader objects with batch size 32, with 32 pairs of tensors representing pairs of images and a tensor of length 32 displaying the labels, one for the training data, one for the testing data and another one for the validation data.

```
Num train dataloaders:62
Num val dataloaders:7
Num test dataloaders:32
```

<u>Model Architecture</u>:

We used the following architecture that was presented in the article:



This architecture has:

- 4 layers of convolution with ReLU as activation function.
- 3 Max Pooling layers with filter size 2x2.
- 2 fully-connected layer with Sigmoid as activation function.

Configurations:

We used a batch sizes of 32 and 64 in our experiments.

The weights were initialized as described in the article: STD=0.01 and Mean=0.

We also added Batch normalization to our architecture after each Max pooling layer and defined the max epochs when training the model to 200.

The Loss function we used is Binary Cross Entropy because the classification of this model is binary.

In each epoch we decreased the learning rate by 1% as described in the article, using lr_scheduler function.

We set the early stopping criteria to stop the training if 2 following epochs don't show improvement for 20 epochs as described in the article:

```
if abs(val_loss[-1]-val_loss[-2])<0.0001 or abs(val_loss[-1]-val_loss[-2])>0.1:
    print("Early stopping")
    break
```

The first condition is to check whether we have reached the convergence.

The second condition is to check whether the loss is in a significant upward trend within

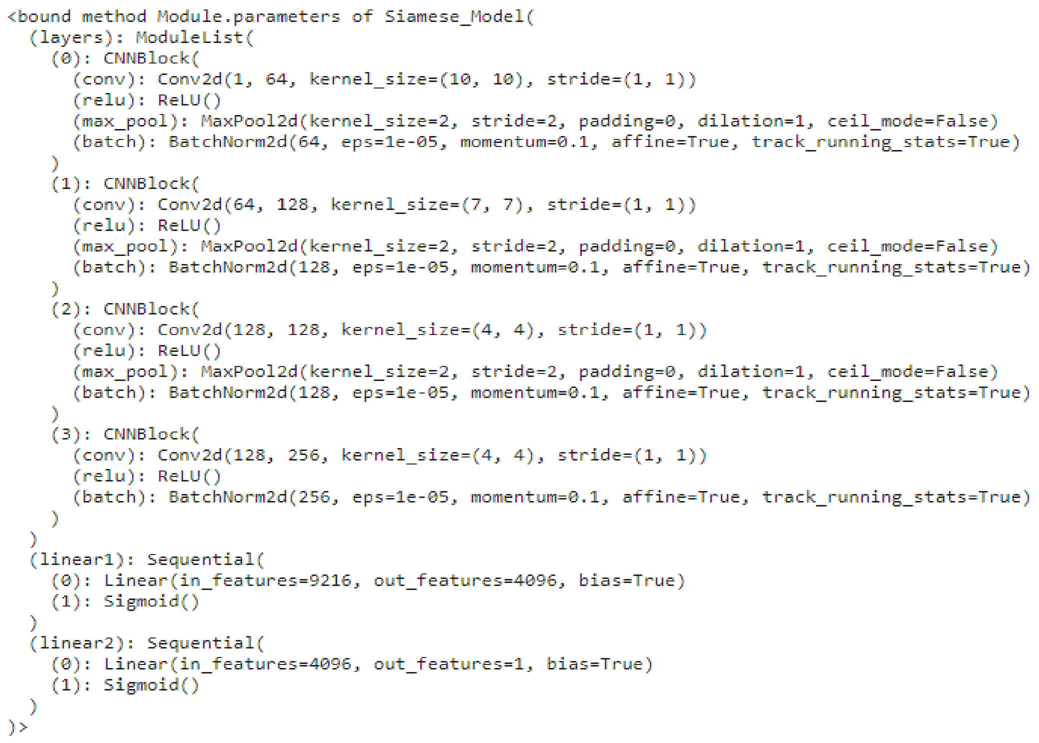
20 epochs.

<u>Experiments</u>:

Experiment 1:

Article's architecture without Dropout using SGD optimizer.

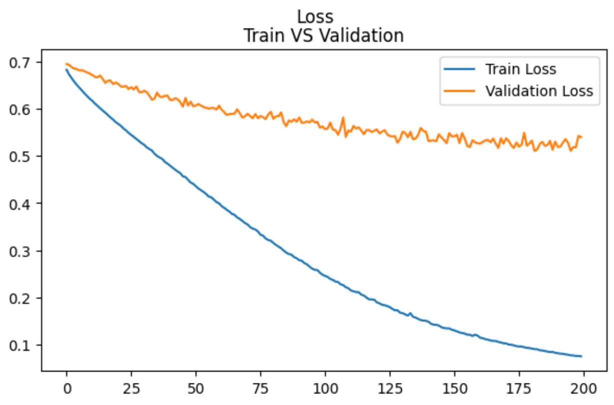
Model parameters:

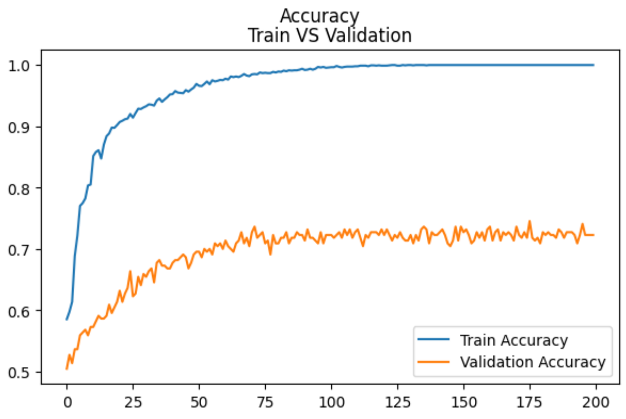


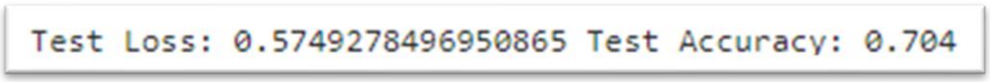
```
<bound method Module.parameters of Siamese_Model(
  (layers): ModuleList(
    (0): CNNBlock(
      (conv): Conv2d(1, 64, kernel_size=(10, 10), stride=(1, 1))
      (relu): ReLU()
      (max_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (batch): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
    (1): CNNBlock(
      (conv): Conv2d(64, 128, kernel_size=(7, 7), stride=(1, 1))
      (relu): ReLU()
      (max_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (batch): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
    (2): CNNBlock(
      (conv): Conv2d(128, 128, kernel_size=(4, 4), stride=(1, 1))
      (relu): ReLU()
      (max_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (batch): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
    (3): CNNBlock(
      (conv): Conv2d(128, 256, kernel_size=(4, 4), stride=(1, 1))
      (relu): ReLU()
      (batch): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (linear1): Sequential(
    (0): Linear(in_features=9216, out_features=4096, bias=True)
    (1): Sigmoid()
  )
  (linear2): Sequential(
    (0): Linear(in_features=4096, out_features=1, bias=True)
    (1): Sigmoid()
  )
)>
```

Results:

```
Epoch: 196, Validation Loss: 0.5186297819018364, Validation Accuracy: 0.7227272727272728, Train_loss: 0.07729876834538675
Epoch: 197, Validation Loss: 0.5182714983820915, Validation Accuracy: 0.7227272727272728, Train_loss: 0.07624861910458534
Epoch: 198, Validation Loss: 0.5425518900156021, Validation Accuracy: 0.7227272727272728, Train_loss: 0.07632652165428284
Epoch: 199, Validation Loss: 0.5397494435310364, Validation Accuracy: 0.7227272727272728, Train_loss: 0.07545865471324613
Total time (sec): 2915.5557672977448
Done!
```







```
Test Loss: 0.5749278496950865 Test Accuracy: 0.704
```

Experiment 2:

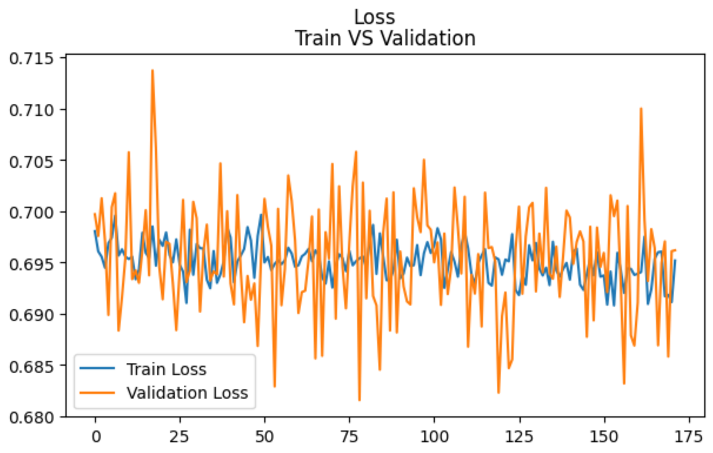
Article's architecture with Dropout using SGD optimizer.

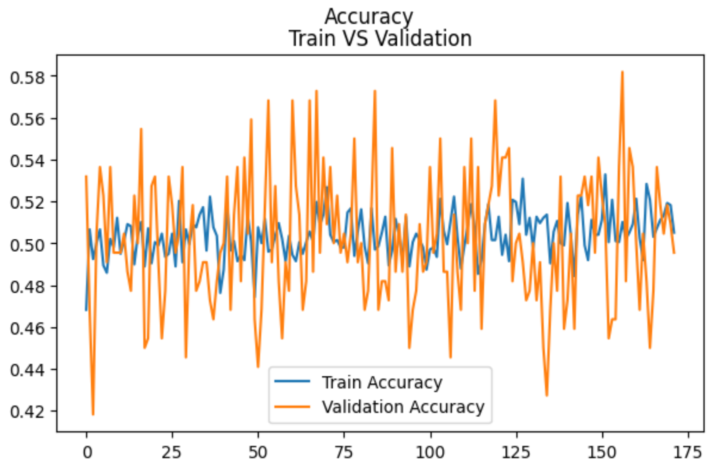
Model parameters:

```
<bound method Module.parameters of Siamese_Model(
  (layers): ModuleList(
    (0): CNNBlock(
      (conv): Conv2d(1, 64, kernel_size=(10, 10), stride=(1, 1))
      (relu): ReLU()
      (max_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (batch): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
    (1): CNNBlock(
      (conv): Conv2d(64, 128, kernel_size=(7, 7), stride=(1, 1))
      (relu): ReLU()
      (max_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (batch): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
    (2): CNNBlock(
      (conv): Conv2d(128, 128, kernel_size=(4, 4), stride=(1, 1))
      (relu): ReLU()
      (max_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (batch): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
    (3): CNNBlock(
      (conv): Conv2d(128, 256, kernel_size=(4, 4), stride=(1, 1))
      (relu): ReLU()
      (batch): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
  )
  (linear1): Sequential(
    (0): Linear(in_features=9216, out_features=4096, bias=True)
    (1): Sigmoid()
  )
  (linear2): Sequential(
    (0): Linear(in_features=4096, out_features=1, bias=True)
    (1): Sigmoid()
  )
)>
```

Results:

```
Epoch: 168, Validation Loss: 0.6970653831958771, Validation Accuracy: 0.5045454545454545, Train_loss: 0.6917108258893413
Epoch: 169, Validation Loss: 0.6858294904232025, Validation Accuracy: 0.5181818181818182, Train_loss: 0.6918564669547542
Epoch: 170, Validation Loss: 0.696111723780632, Validation Accuracy: 0.509090909090909, Train_loss: 0.6911409420351828
Epoch: 171, Validation Loss: 0.6961953938007355, Validation Accuracy: 0.4954545454545455, Train_loss: 0.6951964382202395
Early stopping
Total time (sec): 2504.907879590988
Done!
```





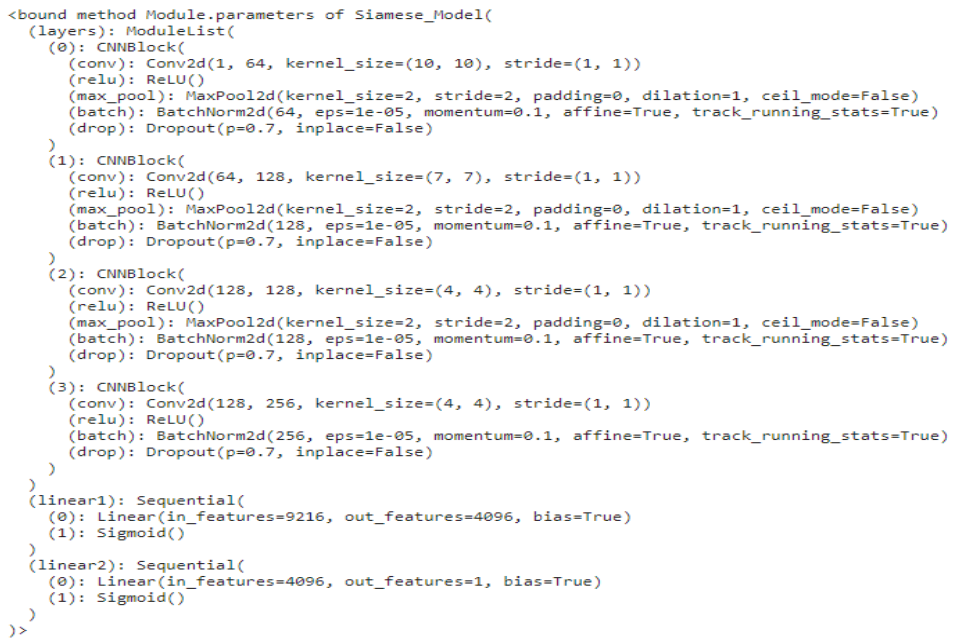
```
Test Loss: 0.6960671581327915 Test Accuracy: 0.493
```

To overcome overfit, we add dropout and the training time reached less epochs to 171, the time was reduced by 411sec. Without dropout, we got maximum epochs and got overfit. In addition, the loss in the validation set gets higher because dropout from 0.539 to 0.696, and the accuracy in validation set gets smaller from 0.722 to 0.495, but we still decided to use Dropout because we want to overcome overfit. The original architecture is very large, and given only 1980 pairs for training, the network learned the training set quickly, leading to overfit. As well, test loss increases to 0.696 from 0.574 and test accuracy decrease from 0.704 to 0.493.

Experiment 3:

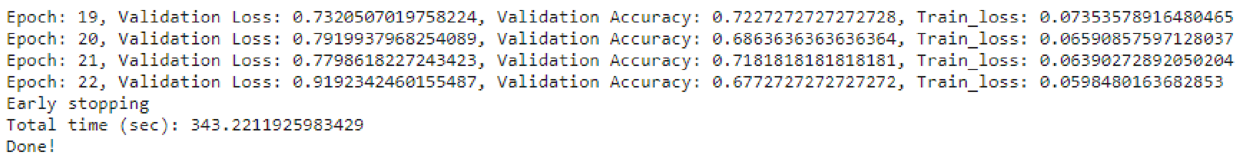
Article's architecture with Dropout using AdamW optimizer.

Model parameters:

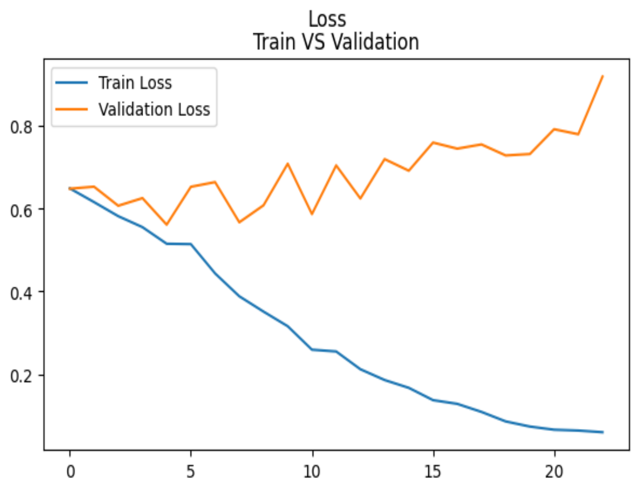


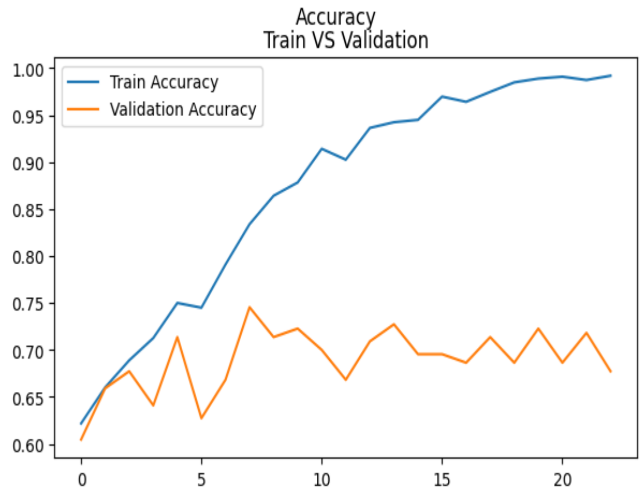
```
<bound method Module.parameters of Siamese_Model(
  (layers): ModuleList(
    (0): CNNBlock(
      (conv): Conv2d(1, 64, kernel_size=(10, 10), stride=(1, 1))
      (relu): ReLU()
      (max_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (batch): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
    (1): CNNBlock(
      (conv): Conv2d(64, 128, kernel_size=(7, 7), stride=(1, 1))
      (relu): ReLU()
      (max_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (batch): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
    (2): CNNBlock(
      (conv): Conv2d(128, 128, kernel_size=(4, 4), stride=(1, 1))
      (relu): ReLU()
      (max_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (batch): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
    (3): CNNBlock(
      (conv): Conv2d(128, 256, kernel_size=(4, 4), stride=(1, 1))
      (relu): ReLU()
      (batch): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
  )
  (linear1): Sequential(
    (0): Linear(in_features=9216, out_features=4096, bias=True)
    (1): Sigmoid()
  )
  (linear2): Sequential(
    (0): Linear(in_features=4096, out_features=1, bias=True)
    (1): Sigmoid()
  )
)>
```

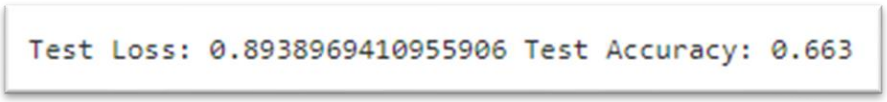
Results:



```
Epoch: 19, Validation Loss: 0.7320507019758224, Validation Accuracy: 0.7227272727272728, Train_loss: 0.07353578916480465
Epoch: 20, Validation Loss: 0.7919937968254089, Validation Accuracy: 0.6863636363636364, Train_loss: 0.06590857597128037
Epoch: 21, Validation Loss: 0.7798618227243423, Validation Accuracy: 0.7181818181818181, Train_loss: 0.06390272892050204
Epoch: 22, Validation Loss: 0.9192342460155487, Validation Accuracy: 0.6772727272727272, Train_loss: 0.0598480163682853
Early stopping
Total time (sec): 343.2211925983429
Done!
```







```
Test Loss: 0.8938969410955906 Test Accuracy: 0.663
```

For optimizers (SGD and AdamW), we see big differences with the training epochs reached 22 epochs instead of 171 epochs and the time reduced by 2161.6 sec. In SGD, the loss in the validation set seems constant already after 60 epochs, but apparently the loss differences between the epochs were not smaller than 0.0001, so the training did not stop. In AdamW, the loss increased right at the beginning but apparently the rise was not big enough for the training to stop.

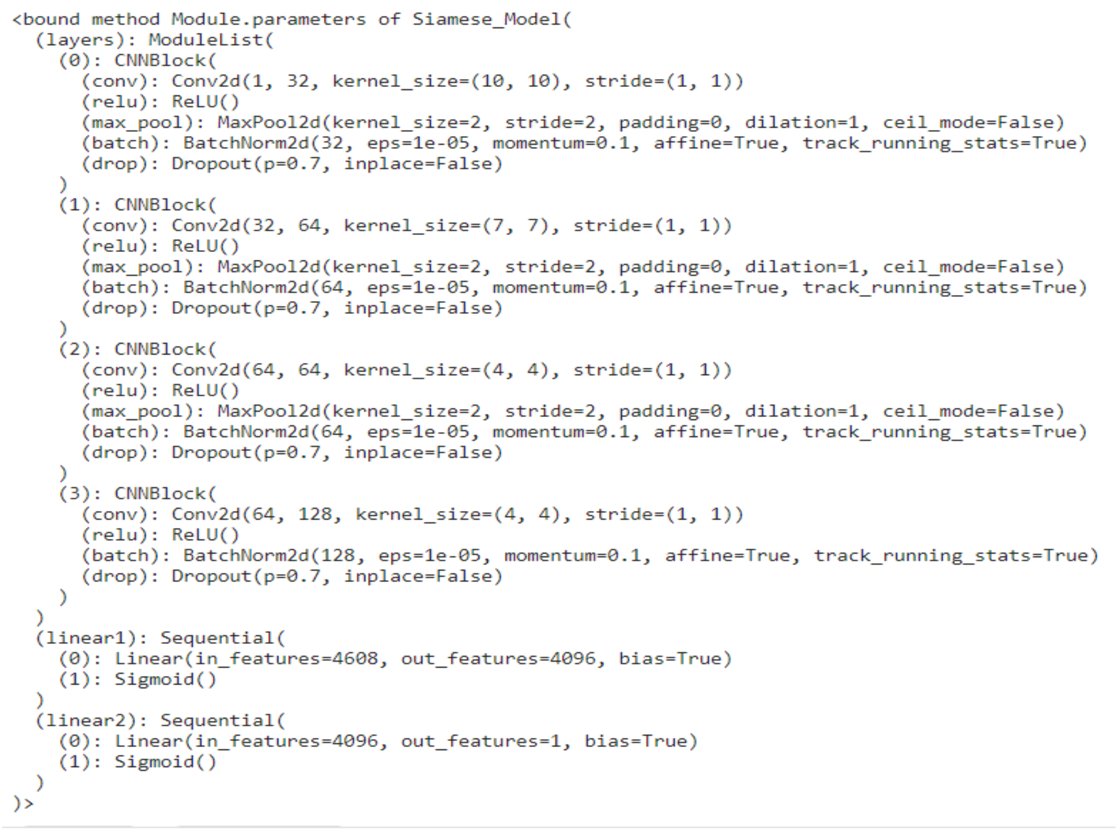
AdamW optimizer gave us higher accuracy than with dropout with SGD from 0.495 to 0.6772 on the validation set and despite the increase of the loss on validation set, we decided to use AdamW in all of the next experiments. As we know from lectures the best optimizer for most of the cases is AdamW.

As well, test loss increases to 0.893 from 0.696 and test accuracy increases from 0.493 to 0.663. To get over loss validation increase problem we had to significantly reduce the number of parameters in each layer.

Experiment 4:

Article's architecture with reduced channels on convolution layers (64 to 32, 128 to 64 and 256 to 128) with Dropout using Adam optimizer.

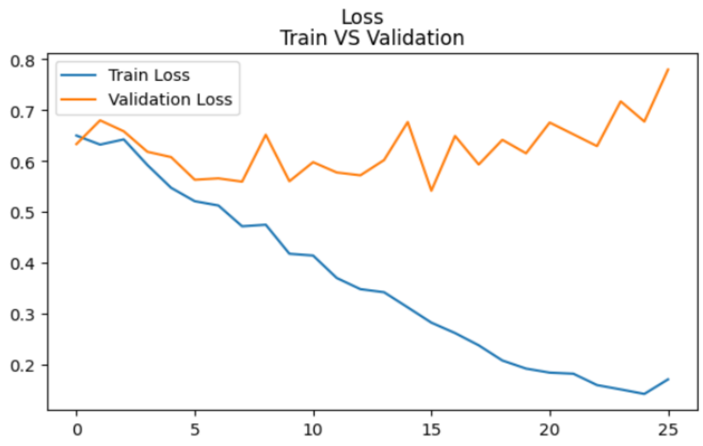
Model parameters:

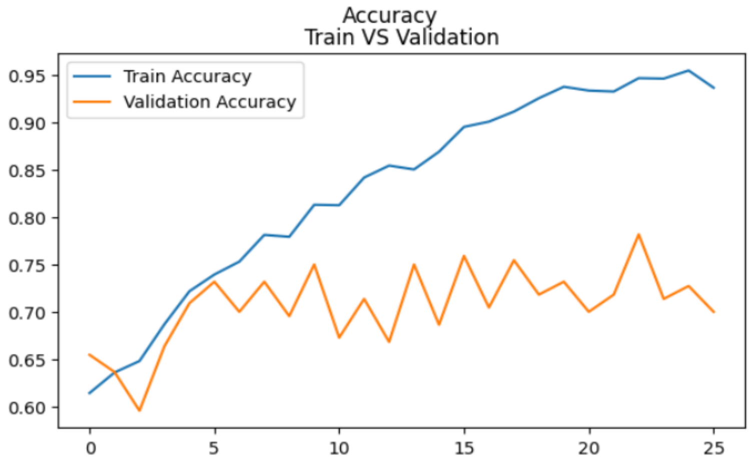


```
<bound method Module.parameters of Siamese_Model(
  (layers): ModuleList(
    (0): CNNBlock(
      (conv): Conv2d(1, 32, kernel_size=(10, 10), stride=(1, 1))
      (relu): ReLU()
      (max_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (batch): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
    (1): CNNBlock(
      (conv): Conv2d(32, 64, kernel_size=(7, 7), stride=(1, 1))
      (relu): ReLU()
      (max_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (batch): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
    (2): CNNBlock(
      (conv): Conv2d(64, 64, kernel_size=(4, 4), stride=(1, 1))
      (relu): ReLU()
      (max_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (batch): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
    (3): CNNBlock(
      (conv): Conv2d(64, 128, kernel_size=(4, 4), stride=(1, 1))
      (relu): ReLU()
      (batch): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
  )
  (linear1): Sequential(
    (0): Linear(in_features=4608, out_features=4096, bias=True)
    (1): Sigmoid()
  )
  (linear2): Sequential(
    (0): Linear(in_features=4096, out_features=1, bias=True)
    (1): Sigmoid()
  )
)>
```

Results:

```
Epoch: 20, Validation Loss: 0.6756856068968773, Validation Accuracy: 0.7, Train_loss: 0.18456032391517393
Epoch: 21, Validation Loss: 0.6525935158133507, Validation Accuracy: 0.7181818181818181, Train_loss: 0.182441592456833
Epoch: 22, Validation Loss: 0.6294110864400864, Validation Accuracy: 0.7818181818181819, Train_loss: 0.16011384658275113
Epoch: 23, Validation Loss: 0.7173004001379013, Validation Accuracy: 0.7136363636363636, Train_loss: 0.1515893474701912
Epoch: 24, Validation Loss: 0.6776975095272064, Validation Accuracy: 0.7272727272727273, Train_loss: 0.14274254489329555
Epoch: 25, Validation Loss: 0.779718205332756, Validation Accuracy: 0.7, Train_loss: 0.1713126606998905
Early stopping
Total time (sec): 340.5604875087738
Done!
```





```
Test Loss: 0.9281479306519032 Test Accuracy: 0.646
```

There is still overfitting and the test accuracy worse from 0.663 to 0.646, but the loss in the training set decreases more slowly. In the validation set, we can see that the loss decreases from 0.919 to 0.779 and accuracy increases from 0.677 to 0.7. To improve the overfitting, we remove one layer from the model.

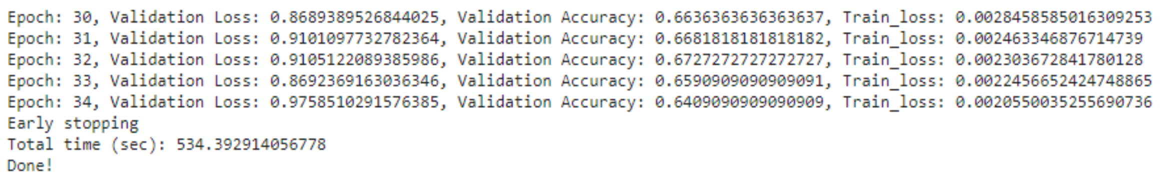
Experiment 5:

Last experiment architecture after removing (64,64) convolutional layer with Dropout using Adam optimizer.

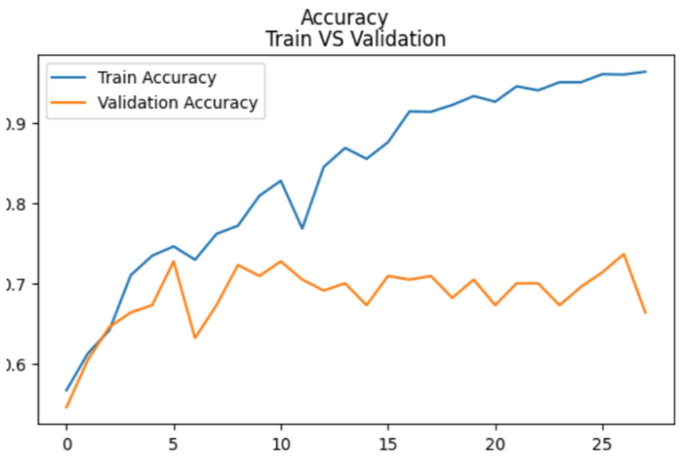
Model parameters

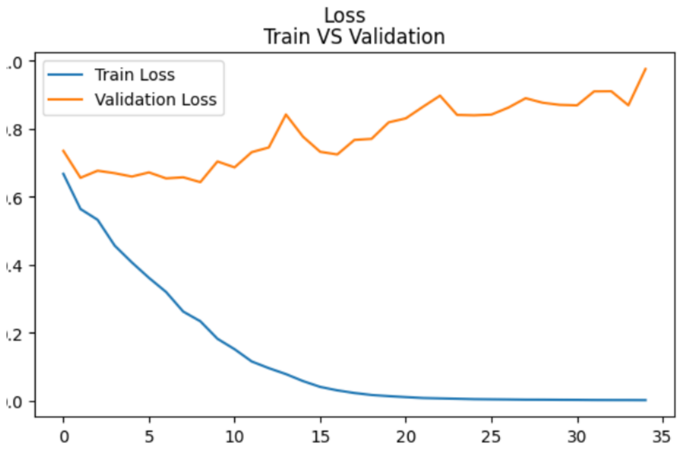
```
<bound method Module.parameters of Siamese_Model(
  (layers): ModuleList(
    (0): CNNBlock(
      (conv): Conv2d(1, 32, kernel_size=(10, 10), stride=(1, 1))
      (relu): ReLU()
      (max_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (batch): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
    (1): CNNBlock(
      (conv): Conv2d(32, 64, kernel_size=(7, 7), stride=(1, 1))
      (relu): ReLU()
      (max_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (batch): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
    (2): CNNBlock(
      (conv): Conv2d(64, 128, kernel_size=(4, 4), stride=(1, 1))
      (relu): ReLU()
      (batch): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
  )
  (linear1): Sequential(
    (0): Linear(in_features=41472, out_features=4096, bias=True)
    (1): Sigmoid()
  )
  (linear2): Sequential(
    (0): Linear(in_features=4096, out_features=1, bias=True)
    (1): Sigmoid()
  )
)>
```

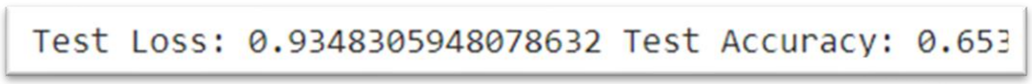
Results:



```
Epoch: 30, Validation Loss: 0.8689389526844025, Validation Accuracy: 0.6636363636363637, Train_loss: 0.0028458585016309253
Epoch: 31, Validation Loss: 0.9101097732782364, Validation Accuracy: 0.6681818181818182, Train_loss: 0.002463346876714739
Epoch: 32, Validation Loss: 0.9105122089385986, Validation Accuracy: 0.6727272727272727, Train_loss: 0.002303672841780128
Epoch: 33, Validation Loss: 0.8692369163036346, Validation Accuracy: 0.6590909090909091, Train_loss: 0.0022456652424748865
Epoch: 34, Validation Loss: 0.9758510291576385, Validation Accuracy: 0.6409090909090909, Train_loss: 0.0020550035255690736
Early stopping
Total time (sec): 534.392914056778
Done!
```







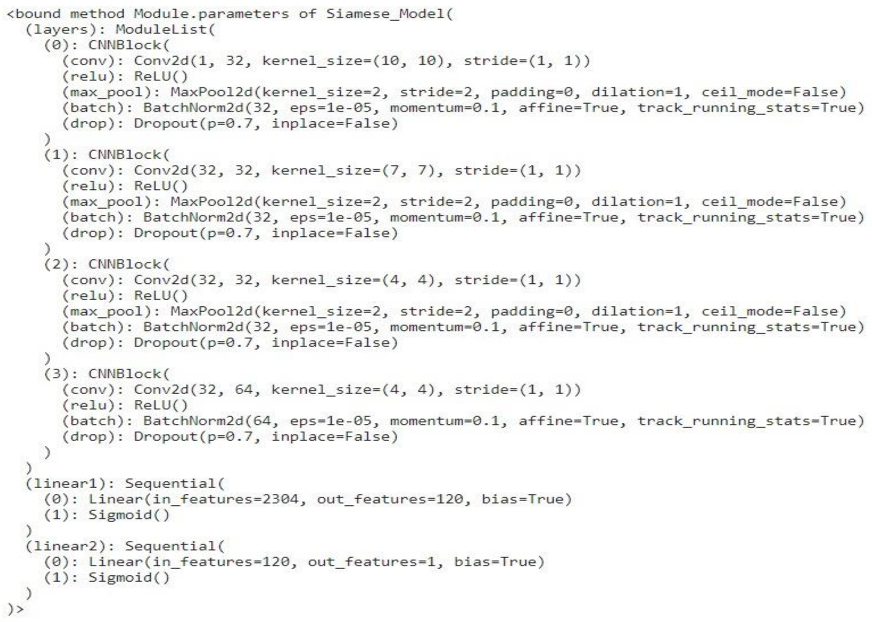
```
Test Loss: 0.9348305948078632 Test Accuracy: 0.653
```

There is still overfitting and the test accuracy worse from 0.663 to 0.653, but the loss in the training set decreases more slowly and getting lower within 34 epochs (more epochs than with conv (64,64)). Unfortunately, in the validation set, we can see that the loss increases from 0.919 to 0.975 and accuracy decreases from 0.677 to 0.64. We understood that this experiment is not good at all, we need to stay this layer maybe with less parameters but this layer improves the architecture. To improve the overfitting, we remove one layer from the mode and add one more layer like we removed in this experiment.

Experiment 6:

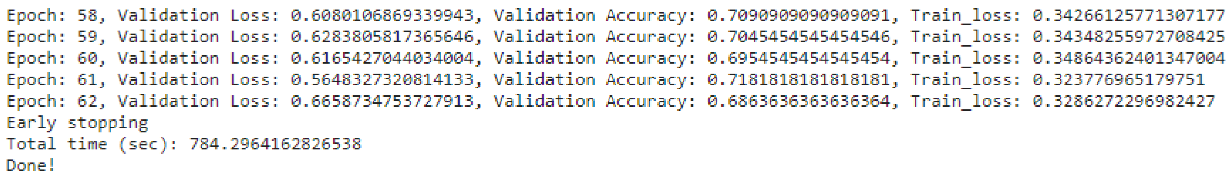
Article's architecture after removing (64,128) convolutional layer and adding (32,32) convolutional layer and changing the middle layer output from 4096 to 120 with Dropout using Adam optimizer.

Model parameters:

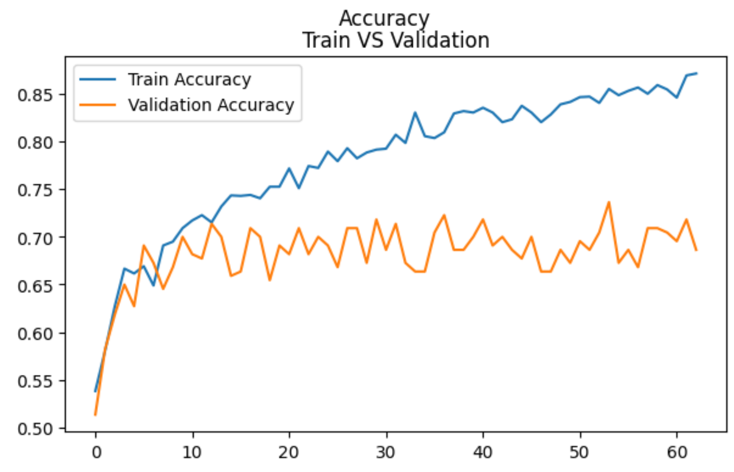


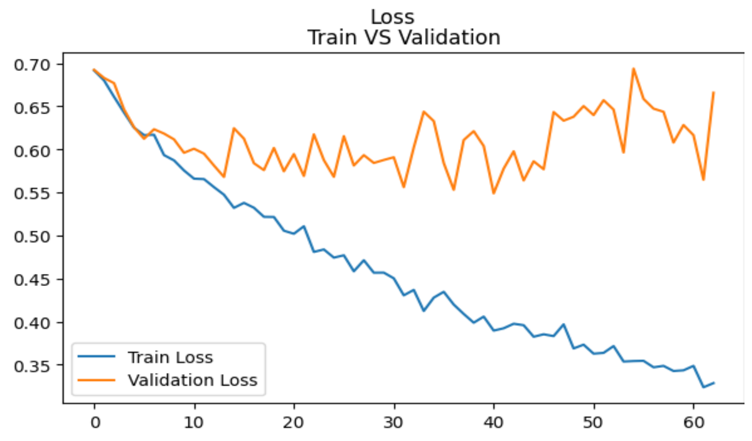
```
<bound method Module.parameters of Siamese_Model(
  (layers): ModuleList(
    (0): CNNBlock(
      (conv): Conv2d(1, 32, kernel_size=(10, 10), stride=(1, 1))
      (relu): ReLU()
      (max_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (batch): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
    (1): CNNBlock(
      (conv): Conv2d(32, 32, kernel_size=(7, 7), stride=(1, 1))
      (relu): ReLU()
      (max_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (batch): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
    (2): CNNBlock(
      (conv): Conv2d(32, 32, kernel_size=(4, 4), stride=(1, 1))
      (relu): ReLU()
      (max_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (batch): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
    (3): CNNBlock(
      (conv): Conv2d(32, 64, kernel_size=(4, 4), stride=(1, 1))
      (relu): ReLU()
      (batch): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
  )
  (linear1): Sequential(
    (0): Linear(in_features=2304, out_features=120, bias=True)
    (1): Sigmoid()
  )
  (linear2): Sequential(
    (0): Linear(in_features=120, out_features=1, bias=True)
    (1): Sigmoid()
  )
)>
```

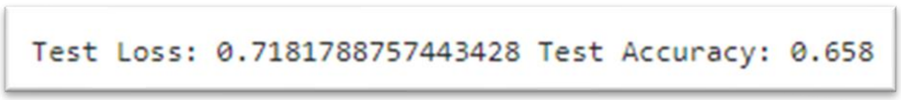
Results:



```
Epoch: 58, Validation Loss: 0.6080106869339943, Validation Accuracy: 0.7090909090909091, Train_loss: 0.34266125771307177
Epoch: 59, Validation Loss: 0.6283805817365646, Validation Accuracy: 0.7045454545454546, Train_loss: 0.34348255972708425
Epoch: 60, Validation Loss: 0.6165427044034004, Validation Accuracy: 0.6954545454545454, Train_loss: 0.34864362401347004
Epoch: 61, Validation Loss: 0.5648327320814133, Validation Accuracy: 0.7181818181818181, Train_loss: 0.323776965179751
Epoch: 62, Validation Loss: 0.6658734753727913, Validation Accuracy: 0.6863636363636364, Train_loss: 0.3286272296982427
Early stopping
Total time (sec): 784.2964162826538
Done!
```







```
Test Loss: 0.7181788757443428 Test Accuracy: 0.658
```

There is still overfitting, but as we expected the test accuracy increases from 0.653 to 0.658 and test loss reduced from 0.934 to 0.718, and the loss in the training set getting lower and stopped within 64 epochs instead 34 epochs.

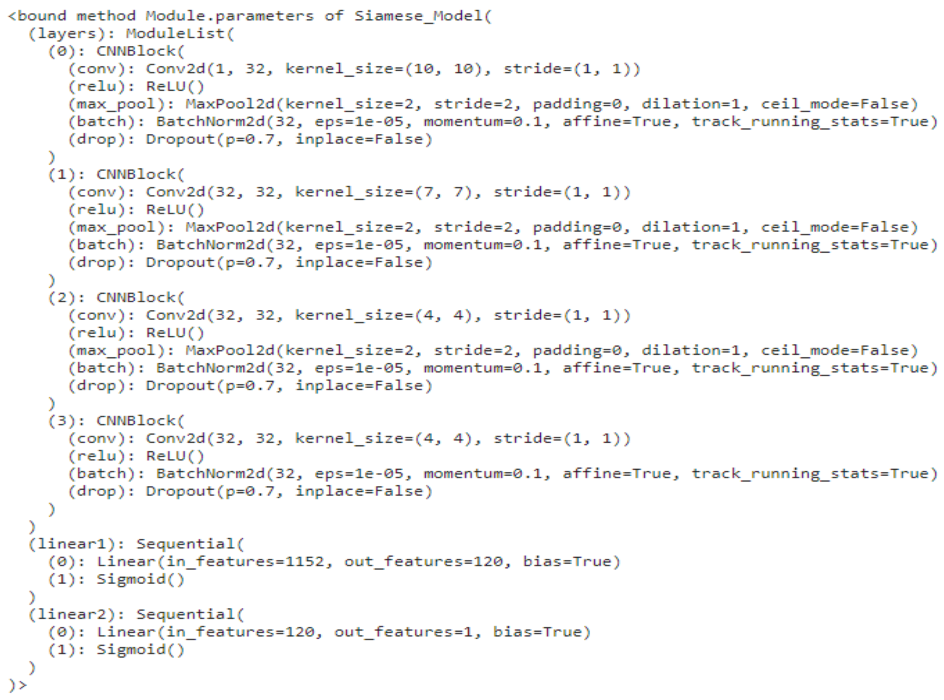
As we expected, in the validation set, we can see that the loss decreased from 0.975 to 0.665 and accuracy increases from 0.64 to 0.686. We understood that this experiment is better than the previous and we keep his way to get out of overfitting. As we said, this layer maybe with less parameters but this layer improves the architecture.

To improve the overfitting, we remove one more layer from the model and add one more layer like we added in this experiment.

Experiment 7:

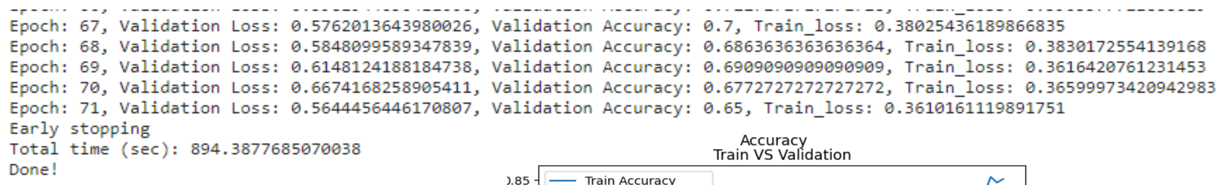
Last experiment architecture after removing (64,128) and (32,64) convolutional layers and adding three (32,32) convolutional layers with Dropout using Adam optimizer.

Model parameters:

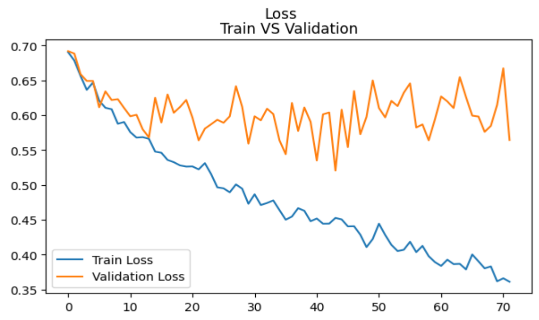


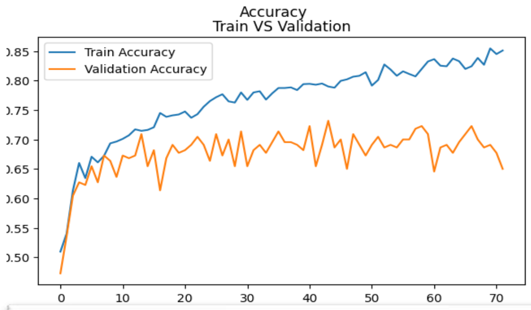
```
<bound method Module.parameters of Siamese_Model(
  (layers): ModuleList(
    (0): CNNBlock(
      (conv): Conv2d(1, 32, kernel_size=(10, 10), stride=(1, 1))
      (relu): ReLU()
      (max_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (batch): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
    (1): CNNBlock(
      (conv): Conv2d(32, 32, kernel_size=(7, 7), stride=(1, 1))
      (relu): ReLU()
      (max_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (batch): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
    (2): CNNBlock(
      (conv): Conv2d(32, 32, kernel_size=(4, 4), stride=(1, 1))
      (relu): ReLU()
      (max_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (batch): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
    (3): CNNBlock(
      (conv): Conv2d(32, 32, kernel_size=(4, 4), stride=(1, 1))
      (relu): ReLU()
      (batch): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
  )
  (linear1): Sequential(
    (0): Linear(in_features=1152, out_features=120, bias=True)
    (1): Sigmoid()
  )
  (linear2): Sequential(
    (0): Linear(in_features=120, out_features=1, bias=True)
    (1): Sigmoid()
  )
)>
```

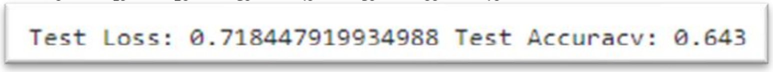
Results:



```
Epoch: 67, Validation Loss: 0.5762013643980026, Validation Accuracy: 0.7, Train_loss: 0.38025436189866835
Epoch: 68, Validation Loss: 0.5848099589347839, Validation Accuracy: 0.6863636363636364, Train_loss: 0.383017255413916 8
Epoch: 69, Validation Loss: 0.6148124188184738, Validation Accuracy: 0.6909090909090909, Train_loss: 0.361642076123145 3
Epoch: 70, Validation Loss: 0.6674168258905411, Validation Accuracy: 0.6772727272727272, Train_loss: 0.36599973420942983
Epoch: 71, Validation Loss: 0.5644456446170807, Validation Accuracy: 0.65, Train_loss: 0.3610161119891751
Early stopping
Total time (sec): 894.3877685070038
Done!
```







```
Test Loss: 0.718447919934988 Test Accuracy: 0.643
```

There is much less overfitting, as we want, we getting out of overfitting but the test accuracy decreases from 0.658 to 0.643 and test loss stays same 0.718, the loss in the training set getting lower and stopped within 71 epochs instead 64 epochs.

As we expected, in the validation set, we can see that the loss decreased from 0.665 to 0.564. We understood that this experiment is better than the previous despite the testing measures and we keep his way to get out of overfitting.

To improve the overfitting, we change the Fully Connected layer.

Experiment 8:

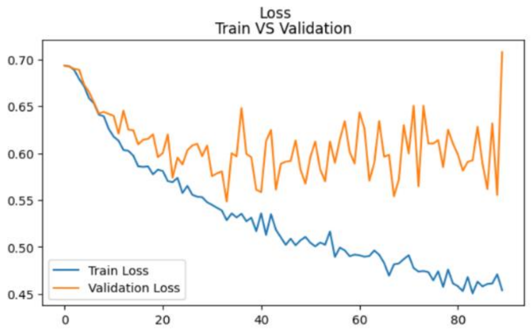
Last experiment architecture after changing the middle layer output from 120 to 32 with Dropout using Adam optimizer.

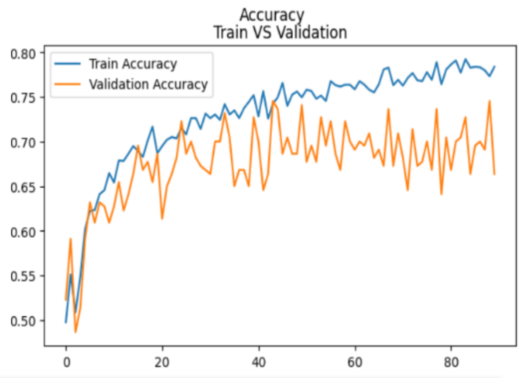
Model parameters:

```
<bound method Module.parameters of Siamese_Model(
  (layers): ModuleList(
    (0): CNNBlock(
      (conv): Conv2d(1, 32, kernel_size=(10, 10), stride=(1, 1))
      (relu): ReLU()
      (max_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (batch): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
    (1): CNNBlock(
      (conv): Conv2d(32, 32, kernel_size=(7, 7), stride=(1, 1))
      (relu): ReLU()
      (max_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (batch): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
    (2): CNNBlock(
      (conv): Conv2d(32, 32, kernel_size=(4, 4), stride=(1, 1))
      (relu): ReLU()
      (max_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (batch): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
    (3): CNNBlock(
      (conv): Conv2d(32, 32, kernel_size=(4, 4), stride=(1, 1))
      (relu): ReLU()
      (batch): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
  )
  (linear1): Sequential(
    (0): Linear(in_features=1152, out_features=32, bias=True)
    (1): Sigmoid()
  )
  (linear2): Sequential(
    (0): Linear(in_features=32, out_features=1, bias=True)
    (1): Sigmoid()
  )
)>
```

Results:

```
Epoch: 85, Validation Loss: 0.589252308011055, Validation Accuracy: 0.6954545454545454, Train_loss: 0.4579052867427949
Epoch: 86, Validation Loss: 0.5616644620895386, Validation Accuracy: 0.7, Train_loss: 0.460533337281750095
Epoch: 87, Validation Loss: 0.6317832767963409, Validation Accuracy: 0.6909090909090909, Train_loss: 0.46098546827993087
Epoch: 88, Validation Loss: 0.5552940517663956, Validation Accuracy: 0.7454545454545455, Train_loss: 0.470828115940094
Epoch: 89, Validation Loss: 0.7077735364437103, Validation Accuracy: 0.6636363636363637, Train_loss: 0.45381263282991224
Early stopping
Total time (sec): 1118.7981300354004
Done!
```





```
Test Loss: 0.7012972086668015 Test Accuracy: 0.641
```

We get out of overfitting, as we wanted, the test accuracy stayed approximately the same 0.641 and test loss stays approximately the same 0.705, the loss in the training set getting lower and stopped within 89 epochs instead 71 epochs.

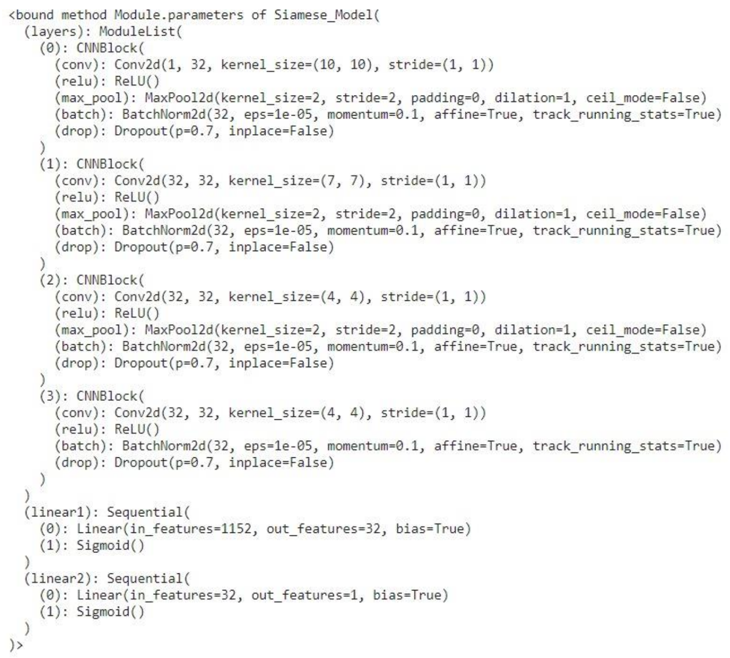
Unfortunately, in the validation set, we can see that the loss gets higher from 0.567 to 0.707 but accuracy increases to 0.663 from 0.65. We understood that this experiment is not work so well but we saved it to get out of overfitting.

To improve the results, we changed the batch size to smaller batch size.

Experiment 9:

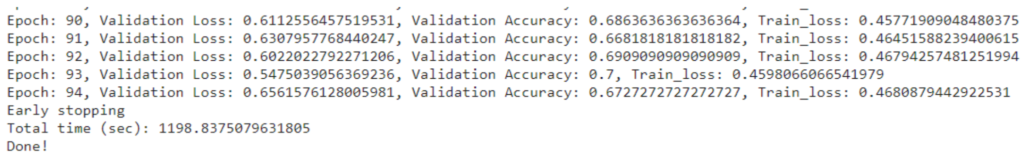
Last experiment architecture with Dropout using Adam optimizer with Batch size of 32 instead of 64.

Model parameters:

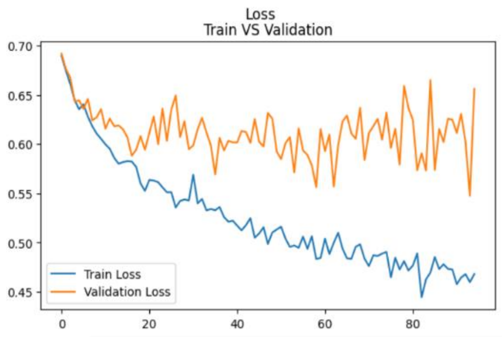


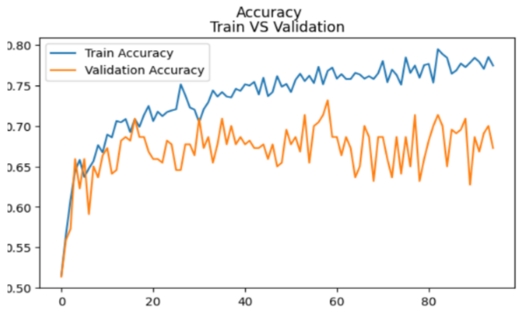
```
<bound method Module.parameters of Siamese_Model(
  (layers): ModuleList(
    (0): CNNBlock(
      (conv): Conv2d(1, 32, kernel_size=(10, 10), stride=(1, 1))
      (relu): ReLU()
      (max_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (batch): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
    (1): CNNBlock(
      (conv): Conv2d(32, 32, kernel_size=(7, 7), stride=(1, 1))
      (relu): ReLU()
      (max_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (batch): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
    (2): CNNBlock(
      (conv): Conv2d(32, 32, kernel_size=(4, 4), stride=(1, 1))
      (relu): ReLU()
      (max_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (batch): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
    (3): CNNBlock(
      (conv): Conv2d(32, 32, kernel_size=(4, 4), stride=(1, 1))
      (relu): ReLU()
      (batch): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
  )
  (linear1): Sequential(
    (0): Linear(in_features=1152, out_features=32, bias=True)
    (1): Sigmoid()
  )
  (linear2): Sequential(
    (0): Linear(in_features=32, out_features=1, bias=True)
    (1): Sigmoid()
  )
)>
```

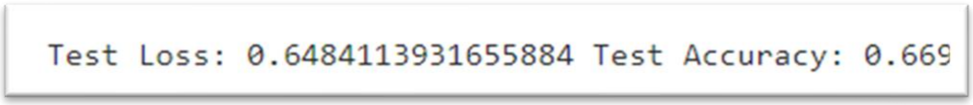
Results:



```
Epoch: 90, Validation Loss: 0.6112556457519531, Validation Accuracy: 0.6863636363636364, Train_loss: 0.45771909048480375
Epoch: 91, Validation Loss: 0.6307957768440247, Validation Accuracy: 0.6681818181818182, Train_loss: 0.46451588239400615
Epoch: 92, Validation Loss: 0.6022022792271206, Validation Accuracy: 0.6909090909090909, Train_loss: 0.46794257481251994
Epoch: 93, Validation Loss: 0.5475039056369236, Validation Accuracy: 0.7, Train_loss: 0.4598066066541979
Epoch: 94, Validation Loss: 0.6561576128005981, Validation Accuracy: 0.6727272727272727, Train_loss: 0.4680879442922531
Early stopping
Total time (sec): 1198.8375079631805
Done!
```







```
Test Loss: 0.6484113931655884 Test Accuracy: 0.669
```

As we wanted, the test accuracy increases from 0.643 to 0.669 and test loss decreases from 0.702 to 0.648, the loss in the training set getting lower and stopped within 94 epochs instead 71 epochs.

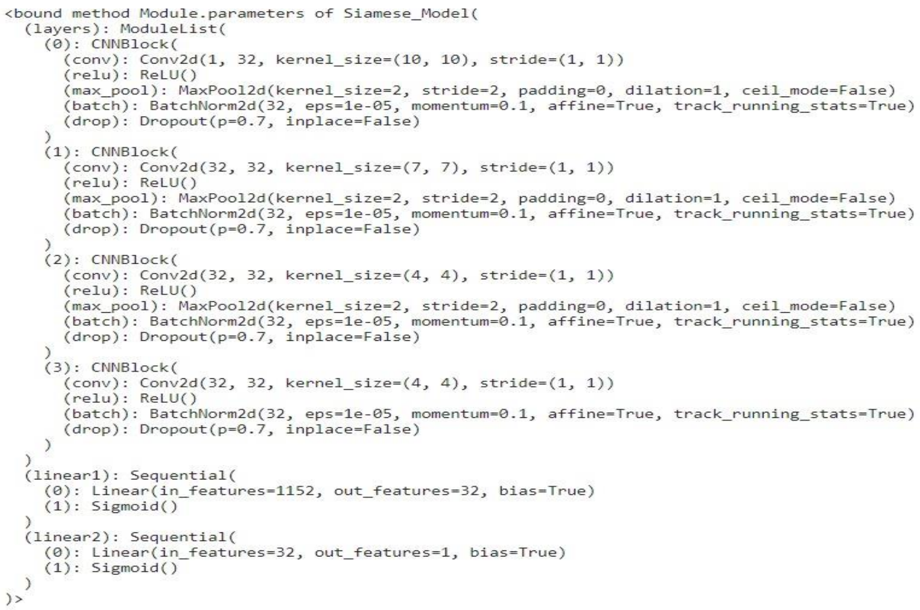
As we expected, in the validation set, we can see that the loss decreased from 0.707 to 0.656 and accuracy increases to 0.672 from 0.663. We understood that this experiment is better than the previous in terms of the testing measures.

To improve more the results, we changed the number of epochs.

Experiment 10:

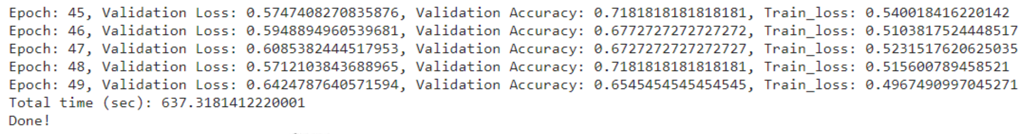
Last experiment architecture with Dropout using Adam optimizer with Batch size of 32 and reducing the number of the max epochs from 200 to 50.

Model parameters:

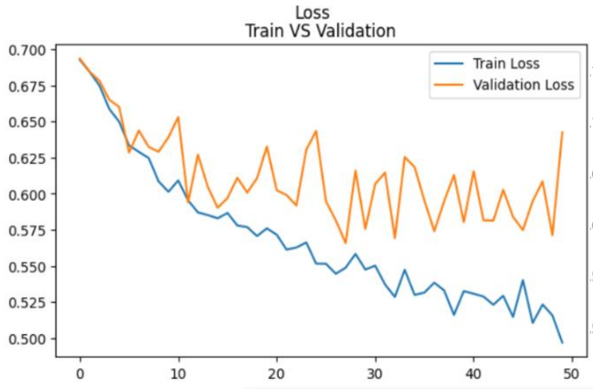


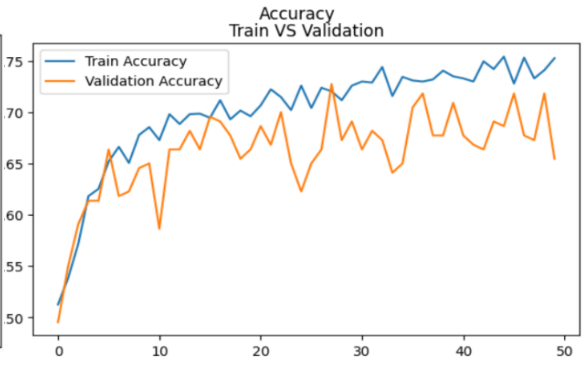
```
<bound method Module.parameters of Siamese_Model(
  (layers): ModuleList(
    (0): CNNBlock(
      (conv): Conv2d(1, 32, kernel_size=(10, 10), stride=(1, 1))
      (relu): ReLU()
      (max_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (batch): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
    (1): CNNBlock(
      (conv): Conv2d(32, 32, kernel_size=(7, 7), stride=(1, 1))
      (relu): ReLU()
      (max_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (batch): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
    (2): CNNBlock(
      (conv): Conv2d(32, 32, kernel_size=(4, 4), stride=(1, 1))
      (relu): ReLU()
      (max_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (batch): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
    (3): CNNBlock(
      (conv): Conv2d(32, 32, kernel_size=(4, 4), stride=(1, 1))
      (relu): ReLU()
      (batch): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (drop): Dropout(p=0.7, inplace=False)
    )
  )
  (linear1): Sequential(
    (0): Linear(in_features=1152, out_features=32, bias=True)
    (1): Sigmoid()
  )
  (linear2): Sequential(
    (0): Linear(in_features=32, out_features=1, bias=True)
    (1): Sigmoid()
  )
)>
```

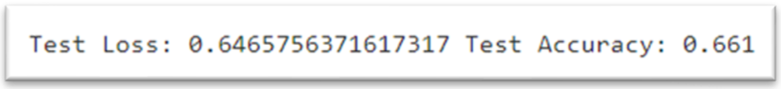
Results:



```
Epoch: 45, Validation Loss: 0.5747408270835876, Validation Accuracy: 0.7181818181818181, Train_loss: 0.540018416220142
Epoch: 46, Validation Loss: 0.5948894960539681, Validation Accuracy: 0.6772727272727272, Train_loss: 0.5103817524448517
Epoch: 47, Validation Loss: 0.6085382444517953, Validation Accuracy: 0.6727272727272727, Train_loss: 0.5231517620625035
Epoch: 48, Validation Loss: 0.5712103843688965, Validation Accuracy: 0.7181818181818181, Train_loss: 0.515600789458521
Epoch: 49, Validation Loss: 0.6424787640571594, Validation Accuracy: 0.6545454545454545, Train_loss: 0.4967490997045271
Total time (sec): 637.3181412220001
Done!
```







```
Test Loss: 0.6465756371617317 Test Accuracy: 0.661
```

Unfortunately, the test accuracy decreases from 0.669 to 0.661 and test loss approximately stay the same 0.646, the loss in the training set getting lower and stopped after maximum epochs (50 epochs).

As we expected, in the validation set, we can see that the loss decreased from 0.656 to 0.642 but the accuracy decreases to 0.663 from 0.654.

We come with conclusion not to take a low maximum epoch to improve the results in this architecture.

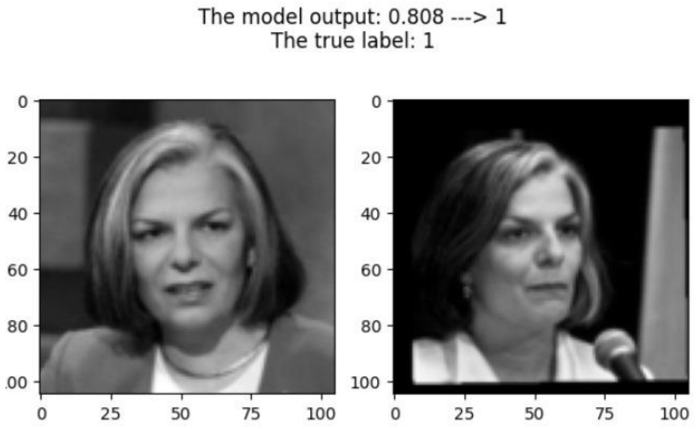
Results Summary:

Experiment/index	Validation loss	Validation accuracy(%)	Train loss	Test loss	Test accuracy(%)	Running time(sec)
1	0.539	72.27	0.0755	0.575	70.4	2915.55
2	0.696	49.55	0.695	0.696	49.3	2504.91
3	0.919	67.73	0.0598	0.894	66.3	343.22
4	0.779	70	0.171	0.928	64.6	340.56
5	0.976	64.09	0.002	0.935	65.3	534.39
6	0.666	68.64	0.329	0.718	65.8	784.29
7	0.564	65	0.361	0.718	64.3	894.39
8	0.708	66.36	0.454	0.701	64.1	1118.79
9	0.656	67.27	0.468	0.648	66.9	1198.84
10	0.642	65.45	0.497	0.647	66.1	637.32

<u>Classification samples</u>:

True Positive:

Classified two images of the same person as the same person (1).

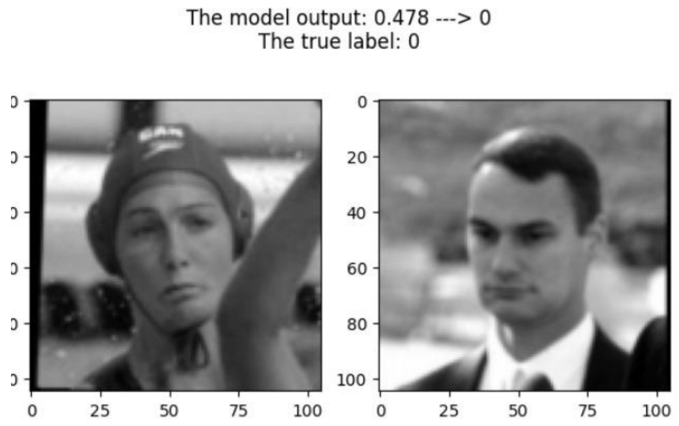


The model classified these two images as the same person with confidence of 80.8% as 1. We can see that both images show women with the same haircut and the same facial expression so we can be sure they are the same woman from first sight.

Although the woman's the direction of the face was different were different in the two pictures, the model still managed to recognize that it was the same figure with a high level of confidence, probably because of the similar features we mentioned.

True Negative:

Classified two images of different persons as different persons (0).



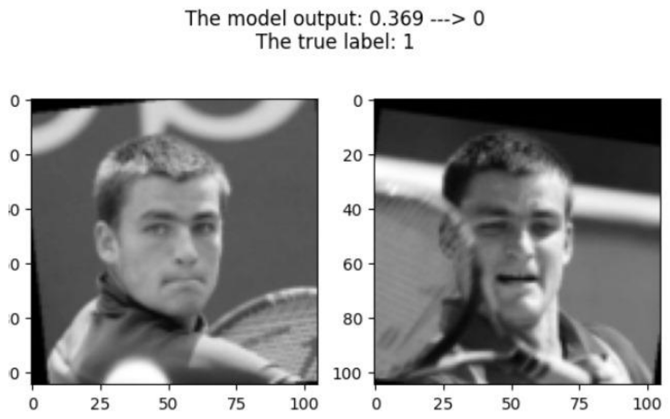
```
The model output: 0.478 ---> 0
The true label: 0
```

These images show different persons that classified as different persons. We can see that one image shows woman and the second image shows a man, moreover, the woman wearing a water pool hat while the man doesn’t so its easy to see the images show different people.

The model distinguished between pictures by their faces, their facial expression and the background are completely different.

False Negative:

Classified two images of the same person (1) as different people (0).



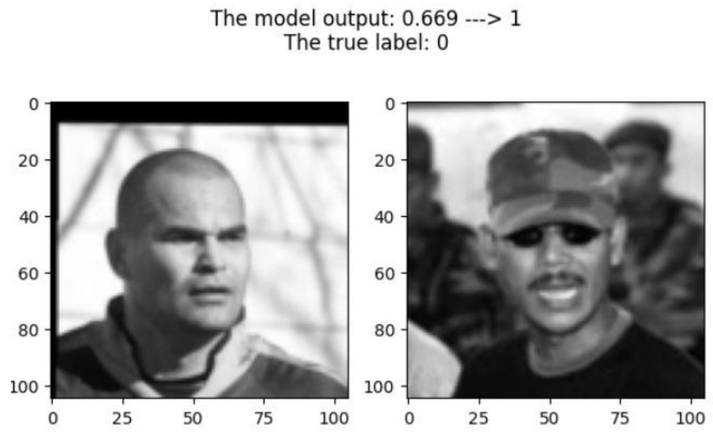
```
The model output: 0.369 ---> 0
The true label: 1
```

We can see that both images show the same person from the same camera angle. We assume that the model misclassified them as different people because in the right image the tennis rocket hides part of the person and his eyes are closed when in the left image his eyes are open.

These might have led the model to an error since it learns features from the entire image.

False Positive:

Classified images of two different people (0) as the same person (1).



In this example we can see two different people that misclassified as the same person. We assume that the model misclassified the images because it seems like this different people have the face features and having the same facial expression in the images. This similarity might have led the model to an error.

<u>Conclusion</u>

The experiments that were performed during the assignment were based on trial and error.

The results show that adding batch normalization and choosing an appropriate optimizer help improve model accuracy, while adding dropout can help reduce overfitting.

Moreover, we found out that the process of finding the best parameters can be very long and the combination of the hyperparameters set for the model impact the final results.

The misclassification examples have shown us that in some cases the wrong classifications were caused due to some details in the background of the images that might have been misleading.