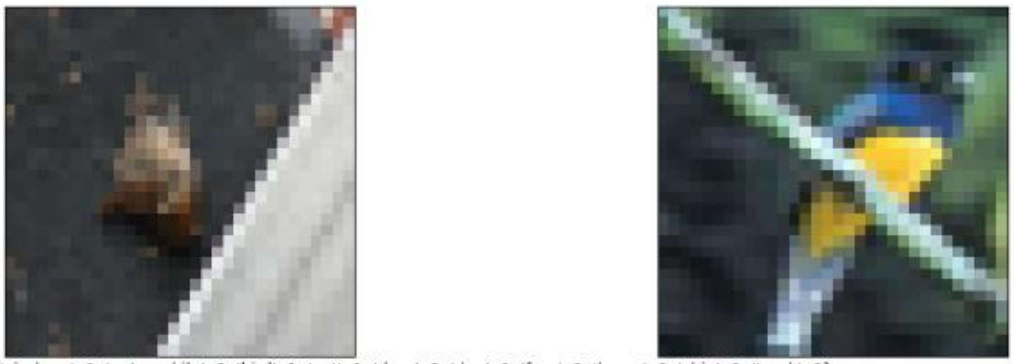
Yaniv Opochinsky

Imubit Challenge:

- **Dataset:** I created a new dataset contains 5K pairs of images from CIFAR10 with their appropriate pair of labels.

  Few examples of the new dataset:

{'airplane': 0, 'automobile': 0, 'bird': 0, 'cat': 0, 'deer': 0, 'dog': 0, 'frog': 1, 'horse': 1, 'ship': 0, 'truck': 0}

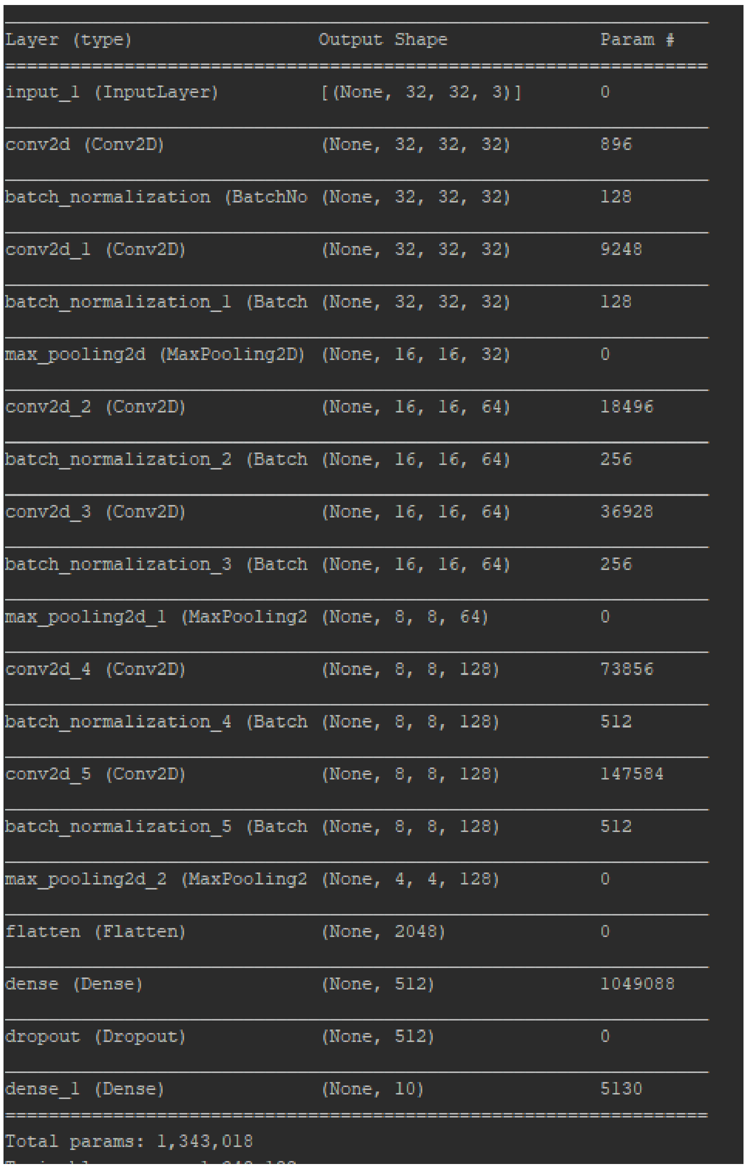

{'airplane': 0, 'automobile': 0, 'bird': 1, 'cat': 1, 'deer': 0, 'dog': 0, 'frog': 0, 'horse': 0, 'ship': 0, 'truck': 0}

{'airplane': 0, 'automobile': 0, 'bird': 1, 'cat': 0, 'deer': 0, 'dog': 0, 'frog': 0, 'horse': 0, 'ship': 0, 'truck': 1}

- For all the following model, I split the train set to train and validation (80\20) and only check the final result (after hyperparameter-tuning) on the test set. I treated the test set as unknown throughout all the training.
  I used data augmentation technique while training to prevent overfitting and to improve the generalization of the model.

- **Network Architecture:** I created a CNN model for the new dataset. The model basically contains 3 building blocks of:

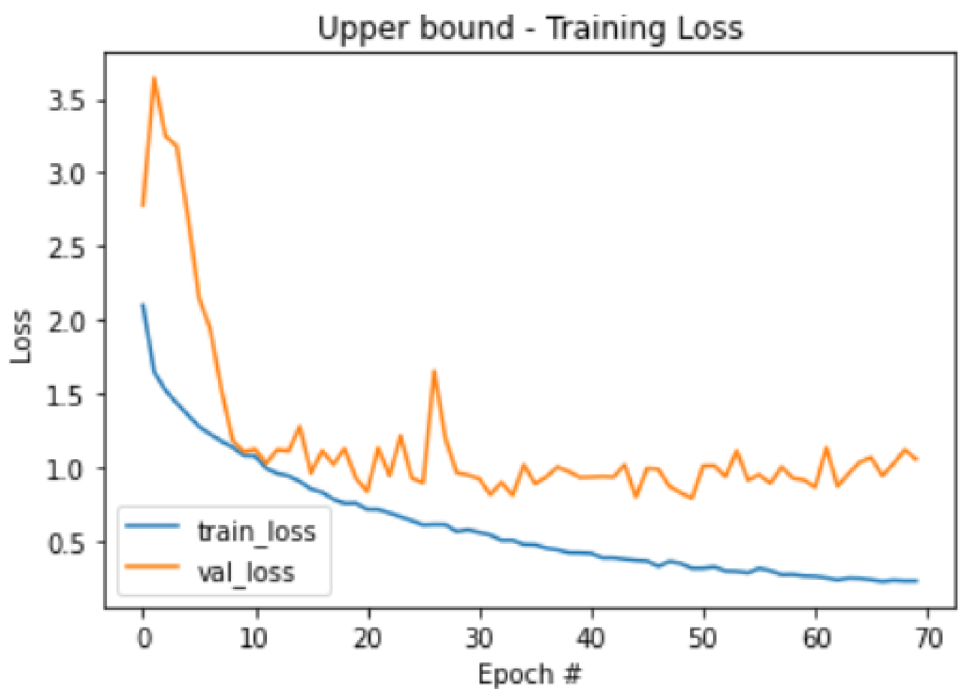
CONV2D -> BN -> CONV2D -> BN -> MaxPooling
On top of it I stick two fully-connected layer. I end the model with 10 units and a softmax activation, which is suitable for the multi-class classification.

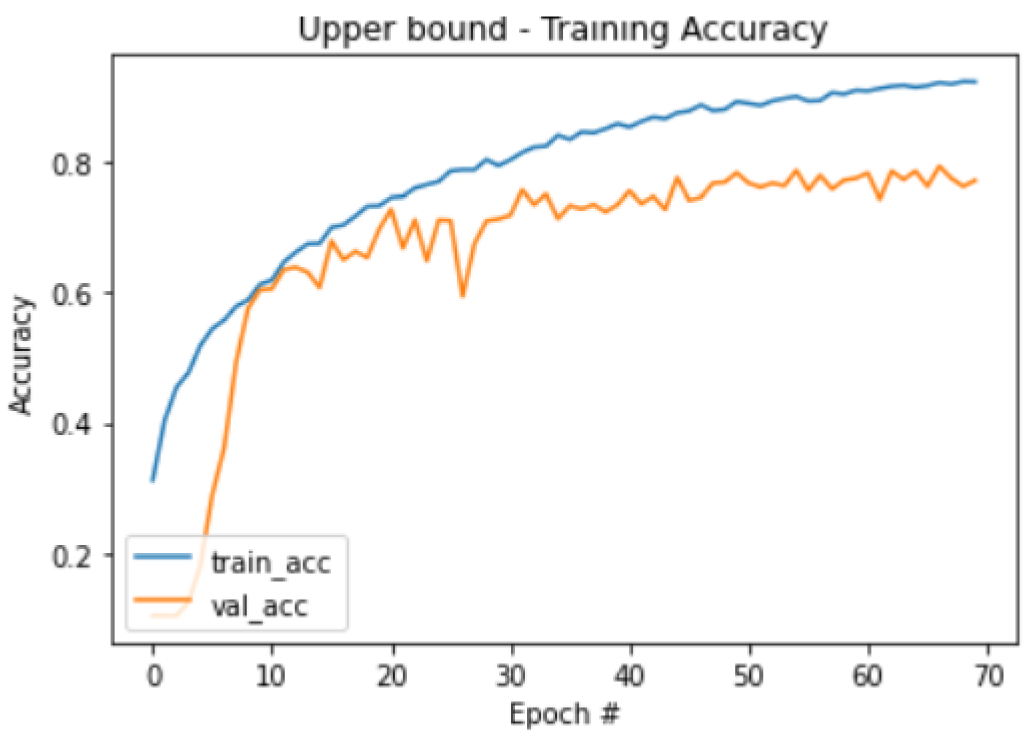


```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 32, 32, 3)]       0

conv2d (Conv2D)              (None, 32, 32, 32)        896

batch_normalization (BatchNo (None, 32, 32, 32)        128

conv2d_1 (Conv2D)            (None, 32, 32, 32)        9248

batch_normalization_1 (Batch (None, 32, 32, 32)        128

max_pooling2d (MaxPooling2D) (None, 16, 16, 32)        0

conv2d_2 (Conv2D)            (None, 16, 16, 64)        18496

batch_normalization_2 (Batch (None, 16, 16, 64)        256

conv2d_3 (Conv2D)            (None, 16, 16, 64)        36928

batch_normalization_3 (Batch (None, 16, 16, 64)        256

max_pooling2d_1 (MaxPooling2 (None, 8, 8, 64)          0

conv2d_4 (Conv2D)            (None, 8, 8, 128)         73856

batch_normalization_4 (Batch (None, 8, 8, 128)         512

conv2d_5 (Conv2D)            (None, 8, 8, 128)         147584

batch_normalization_5 (Batch (None, 8, 8, 128)         512

max_pooling2d_2 (MaxPooling2 (None, 4, 4, 128)         0

flatten (Flatten)            (None, 2048)              0

dense (Dense)                (None, 512)               1049088

dropout (Dropout)            (None, 512)               0

dense_1 (Dense)              (None, 10)                5130
=================================================================
Total params: 1,343,018
```

- I used an early stopping (according the validation loss) to decide when to stop the train.

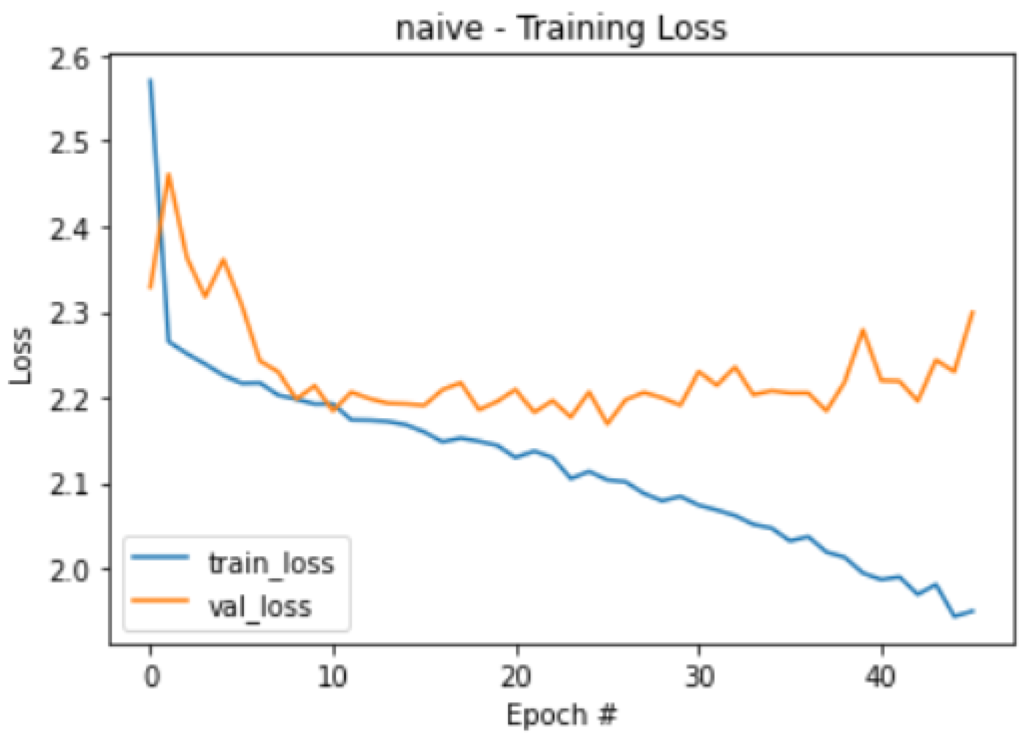
- **Upper Bound Model:** I trained a CNN model on the total 10K images (5K *2) and the **real** labels as upper bound to the expected accuracy in the challenge problem.

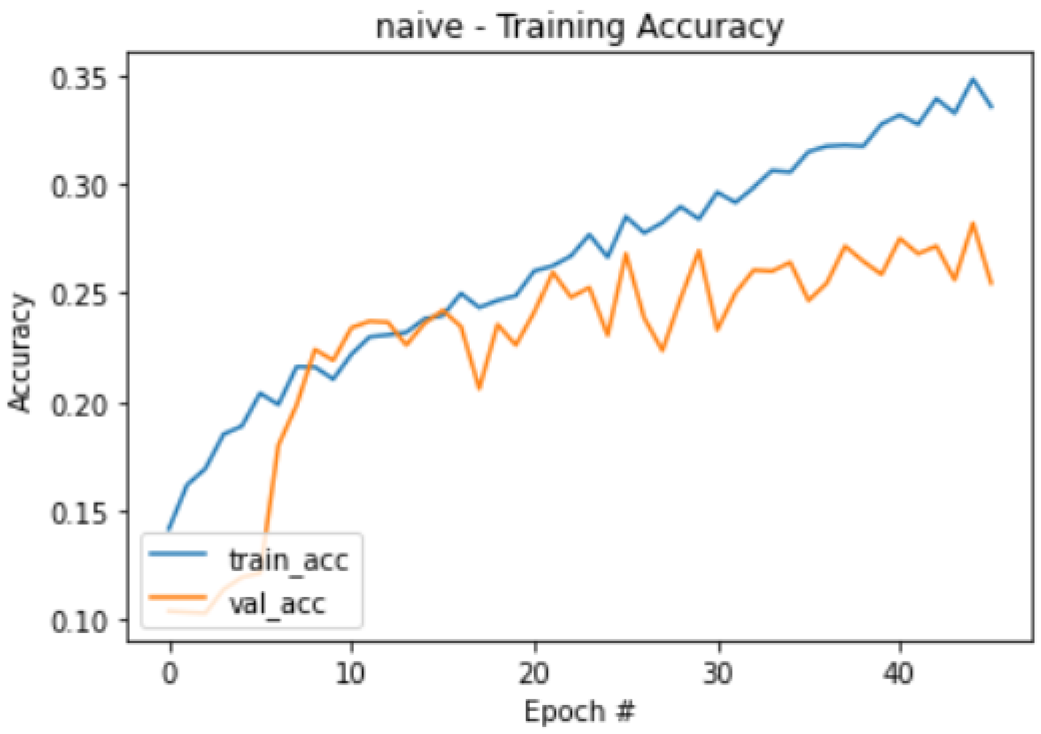




The loss and the accuracy graphs indicate a standard learning and convergence process of the network. The final accuracy (on the test set) is: 0.743

- **Naïve Baseline:** Next, I trained a naïve baseline. This network has same architecture as the upper bound network. It was trained on the pair-based data as follows, randomly assign one of the labels to one of the images. In this way, it is clear that the data has about fifty percent of incorrect labels.





The final accuracy (on the test set) is: 0.473, significantly lower than the accuracy of the upper bound. The network very fast overfitted the train set.

- **Solution:** In order to solve the challenge problem, I used a Siamese network, for the pair of images, with share weights. The loss is calculated twice- for the two possible permutations and the minimal option is chosen.

  Schematic description:

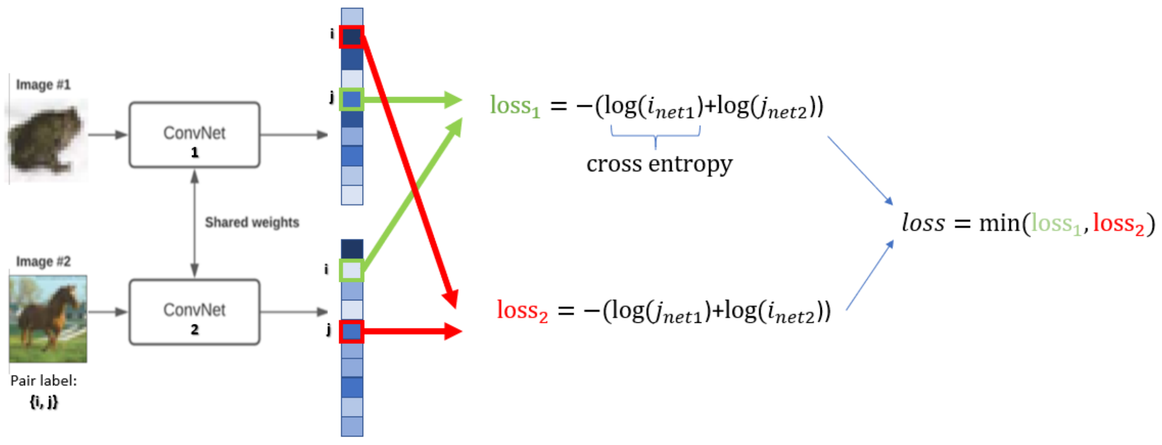
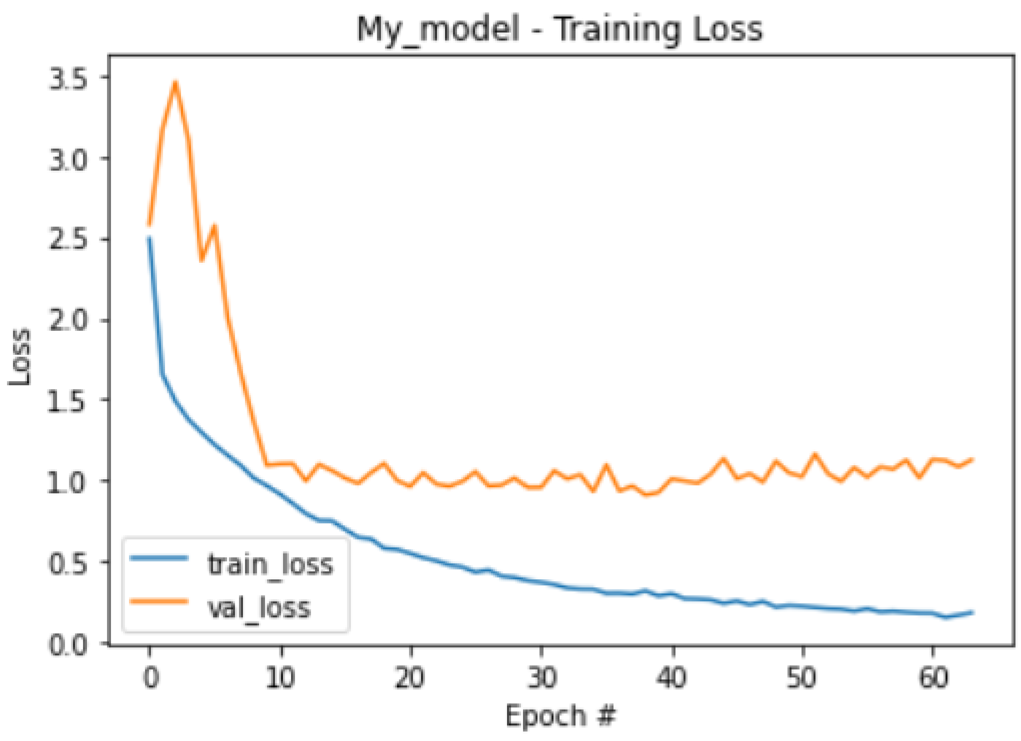
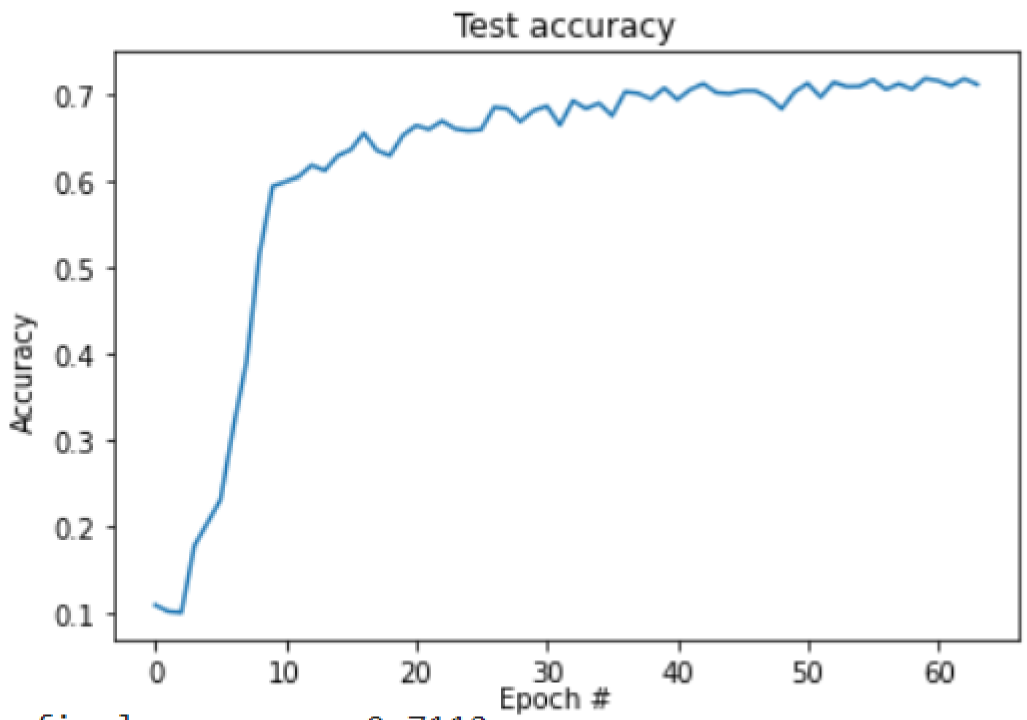


Image #1
ConvNet 1
Shared weights
Image #2
ConvNet 2
Pair label: {i, j}
i
j
i
j
$loss_1 = -(\log(i_{net1})+\log(j_{net2}))$
cross entropy
$loss_2 = -(\log(j_{net1})+\log(i_{net2}))$
$loss = \min(loss_1, loss_2)$



The final accuracy (on the test set) is: 0.711, close to the upper bound. (If I take the validation from the real label, I reach to 0.740, very close to the upper bound)



```
final accuracy  0.7112
```

I try also to initial my model with the weights from the baseline model, but it didn't imporve the final results.

To conclude, my model outperforms the baseline model (0.71 > 0.473) and very close to the upper bound (0.743).

*There are more techniques to improve the overall performance like use more complicated network backbone or transfer learning, but I think it is out of the scope of this challenge.