CAP5404 Deep Learning for Computer Graphics
Dr. Corey Toler-Franklin

Course Project Part II Report

Deep Colorization and Transfer Learning

Submitted by:
Sandeep Jalui, Prateek Behera, Akash Kumar Kondaparthi.

A) How to run code: -

Unzip the project folder. The code is saved in **Final.ipynb** file. Make sure the project is in a CUDA enabled environment. To run the code, open Final.ipynb in a jupyter notebook and then click "run all cells".

In case the code is checked in a local system, a **requirements.txt** file has been included which may be required to make sure the environment is identical to the custom kernel that we used.

```
conda create --name deepcolorizer_env --file requirements.txt
```

Alternatively, if **HiPerGator** is used to check the working of the code, our recommended kernel is **URFC-Python 3.10**.

After running all the cells, the following files and folders will be created or populated.

B) Folders & Files

Folders:

1) **face_images, Gray, ColorfulOriginal –** These are the original provided datasets for the project.
2) **augmented –** Contains all training data of face images including augmented. transformed version used in regressor and colorizer model.
3) **test_folder –** Contains test images of face dataset used in regressor and colorizer model.
4) **a –** a channel of face images.
5) **b –** b channel of face images.
6) **L –** l channel of face images.
7) **color_fruit_train –** Contains train images of NCD data used in transfer model.
8) **color_fruit_test -** Contains test images of NCD data used in transfer model.
9) **output_color –** Contains colorized output for face images coming from colorizer model.
10) **transfer_output –** Contains colorized output for NCD images coming from transfer model.

Files:

1) **regressor.pth** - Trained model for regression model trained on face images predicting mean chrominance.
2) **color.pth** - Trained model for colorizer model trained on face images.
3) **transfer.pth** - Trained model for transfer model trained on NCD images.
4) **regressor_tanh.pth, color_tanh.pth, transfer_tanh.pth** – corresponding files for tanh

C) Loading and Augmenting Data: -

The following figure 1 shows the plots of the first image in the face_image dataset to check how it looks in different form. The first image is the original image, followed by LAB converted, L channel, a channel and b channel respectively.

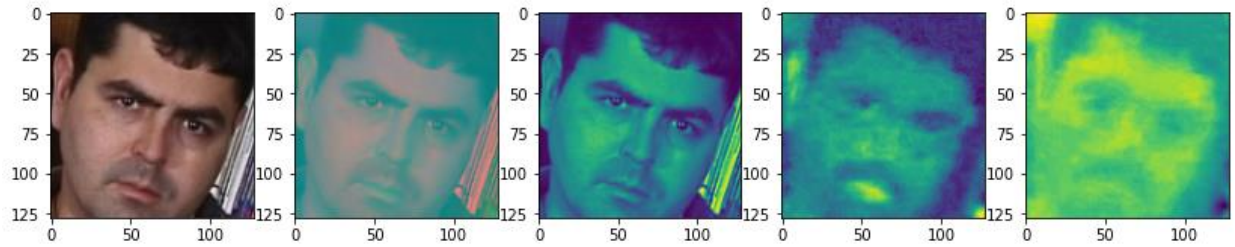

Figure 1: Demo image from the face dataset (Original, LAB, L, a, b)

Augmenting the face image dataset: -

The dataset contains 750 images with each image size 128 x 128. 675 images were considered for training and transformed using 5 augmented techniques. The final augmented image is saved in augmented folder while the remaining 75 images saved in test_folder un-augmented. Five augmentation transforms techiques were RandomCrop, RandomHorizontalFlip, RandomVerticalFlip, RandomRotation, RandomAdjustSharpness.

D) Regressor: -

Architecture: -

The following figure 2 shows the architectural neural network layer of regressor code to predict the mean chrominance of channel a and channel b. The input channel is 1, with 3 hidden channels and 2 output channels for predicting channel a and channel b. There are total 6 modules with each module containing 3 layers, i.e. Conv2d (2D convolution layers), batch normalization and leakyReLU followed by the linear layer in the last. Sigmoid activation function is used to predict the regressor channels. MSEloss function was used to calculate losses in each epoch.

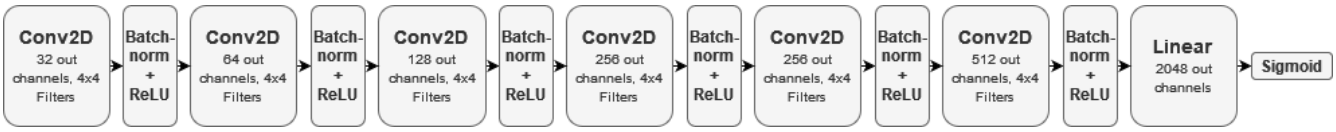

Figure 2: Architecture of the Regressor Model to predict mean chrominance.

The images in the augmented folder are passed through the conversion code to convert it into L, A and B channel. The L channel (grayscale) is given as input to the model and it gives predicted mean chrominance output. With all the hyperparameter tuning, it was tested on test_folder image which are un-augmented and the results were plotted. MSE scores are recorded as shown below.

Hyperparameters: -

The following hyperparameters were chosen after performing Grid Search on Learning rate, no. of Epochs and weight decay.

Table 1: Hyperparameters for Regressor

Parameters	**Values**
Learning Rate	0.001
No. of Epochs	100
Weight decay	1e-4
Optimizer	Adam
Loss	MSE

Results: -

Table 2: MSE Computed for a and b* channels*

Parameters	**MSE**
Channel A	1.4566675e-05
Channel B	6.9435344e-05

Mean Squared Error Loss Plot –

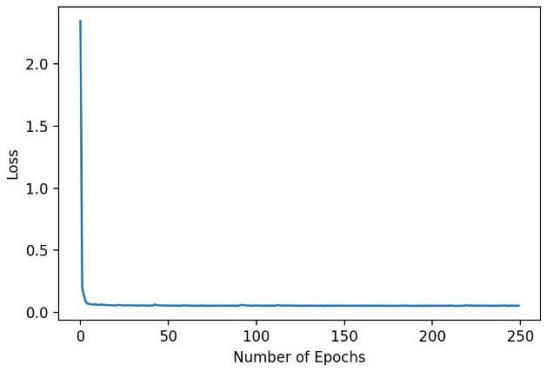

Figure 3: Loss curve vs no. of Epochs (Regressor)

E) Colorizer (Colorization on Face Images): -

Architecture: -

The input to the colorizer model is the L channel (grayscale) from the input face images. The model outputs the predicted chrominance channels i.e., a* and b* channels with the mean chrominances. The colorizer network contains 12 layers split into two sections. The first section is the down-sampler and the second is the up-sampler. The down-sampling section has 6 Conv2d layers (2D convolution layers), each followed by batch-normalization and non-linear ReLU activation. The up-sampling section also has 6 2D Convolution layers with transforms and each layer is followed by batch normalization and ReLU activation except the last layer which has sigmoid as activation function. The figure below shows the model architecture of the colorizer.

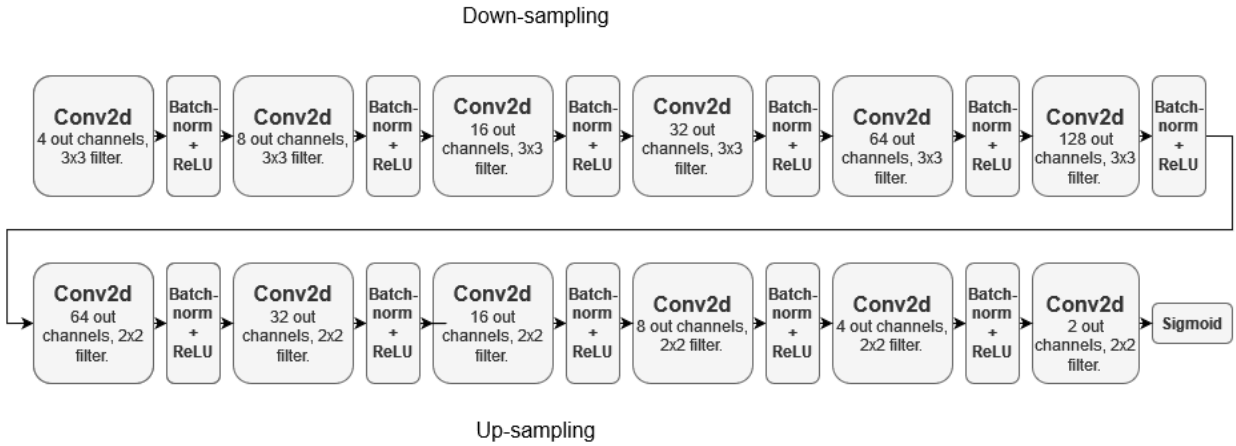

Figure 4: Architecture of the Colorizer Model

The Mean Squared Error is computed between the original image in LAB color space and the predicted image in LAB color space. The LAB color space is obtained by merging the L, a* and b* channels. MSE is used as the loss criterion during training and Adam optimizer is used to obtain optimal weights. We convert the images into RGB to view them and make observations.

Hyperparameters: -

The following hyperparameters were chosen after performing Grid Search on Learning rate, no. of Epochs and weight decay.

Table 3: Hyperparameters for Colorizer

Learning Rate	0.001
No. of Epochs	20
Weight decay	1e-4
Optimizer	Adam
Regularization	Early Stopping
Loss Criterion	MSE

Mean Squared Error Loss Plot -

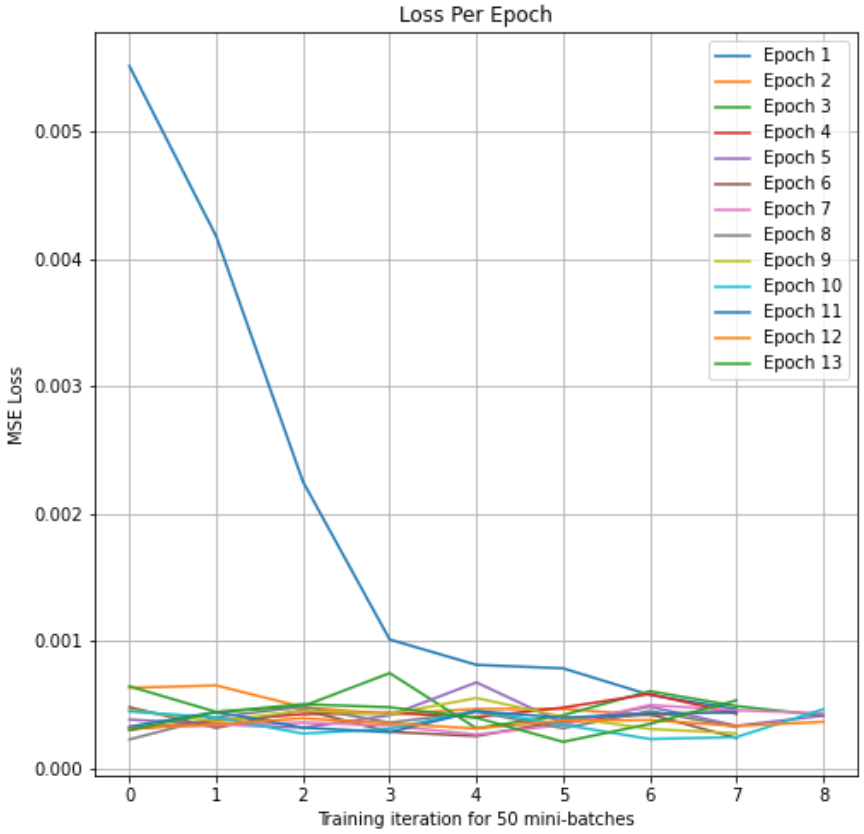

Figure 5: Loss Curve for each Epoch for Colorizer

Since we are using early stopping as a regularization method, the training should stop between 10 to 15 epochs. The plot above shows the MSE loss performance for each epoch while the training on the data.

Results: -

From the figure below, we can see that on comparing the predicted colorized outputs to the original-colored images, that the colorizer model does a good job of accurately colorizing test face images. But we can still see that the predicted outputs are a bit dull in comparison to the original images and the range of colors seem very similar across different images. This may be an indication that the model might be averaging across input images to predict the colors accurately.

Figure 6: Results of Colorizer on Test Face Images

F) Colorization of NCD Dataset using Transfer Learning: -

The trained colorizer model trained on face data is now used to make predictions on the NCD dataset that contains fruits and vegetables images. This approach of using pretrained models on a very slightly different in known as Transfer learning. We split the Colorful original Dataset into train and test sets and applied the same transforms and converts the images into tensors. We then take the pretrained weights and initialize with those weights while training on the smaller NCD dataset and plot the Loss function for each epoch. Finally, the model is used to make predictions on test data.

Hyperparameters: -

The following hyperparameters were chosen after performing Grid Search on Learning rate, no. of Epochs and weight decay.

Table 4: Hyperparameters for Transfer model

Learning Rate	0.001
No. of Epochs	20
Weight decay	1e-4

Optimizer	Adam
Regularization	Early Stopping
Loss Criterion	MSE

Mean Squared Error Loss Plot -

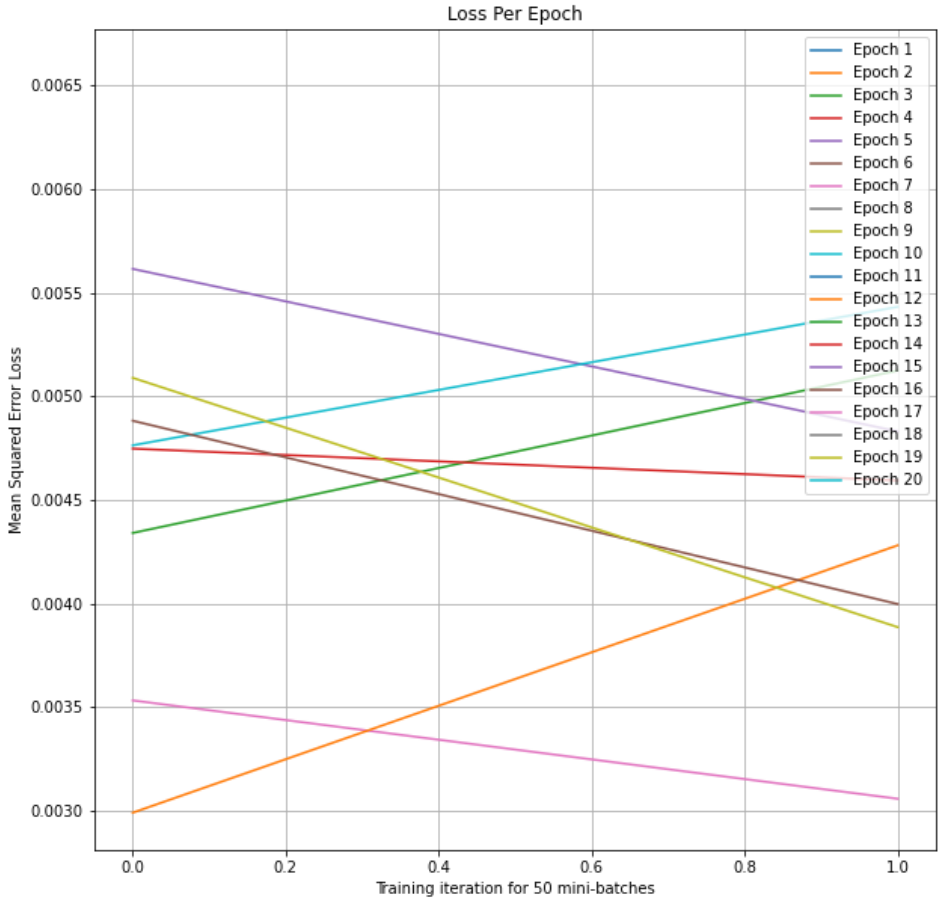

Figure 7: Loss curve for each epoch while training Transfer model

The plot above shows the MSE loss performance for each epoch while training the transfer learning model on the NCD data. We can see here that, since the training was initialized through pretrained weights, only a few steps of training are required to get a respectable performance.

Results: -

We can observe the results in the figure below. The original desired colorized image, the corresponding gray scaled input image, the LAB version of the input image and finally the predicted colorized output are displayed in the figure.

From observing the output in comparison to the original-colored image, we can see that there is a yellow hue in the colorized output. This is due to the initial model being trained on face images where it has learned the hue of the skin colors. Thus, it performs sub optimally on the NCD dataset which has different dataset. This also indicates that the transferred model does not do well when predicting colors that are specific to the NCD dataset and were not observed enough in the face image dataset.

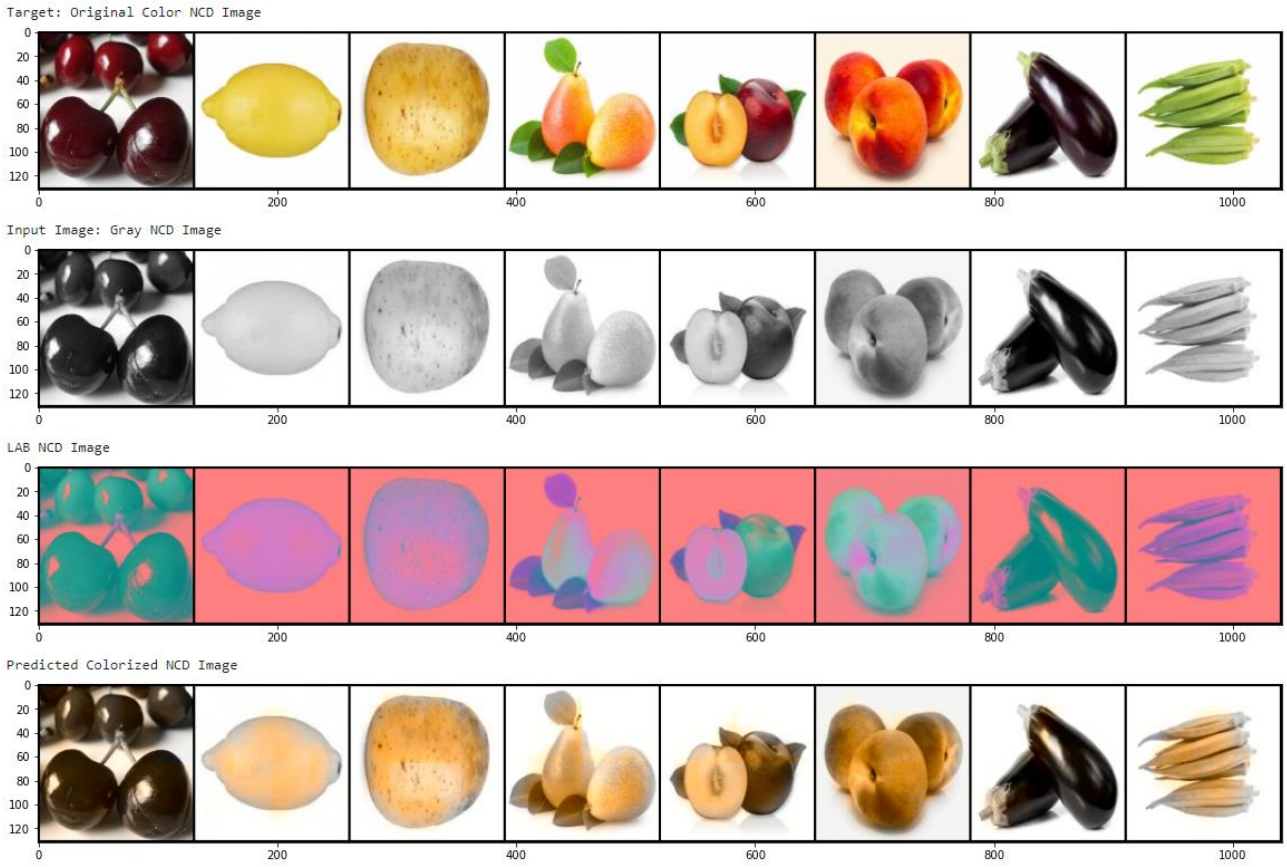

Figure 8: Results of Transfer Model on Test NCD Images

G) Conclusion: -

Regressor model was built on face image dataset which takes l channel as input and predicts the mean chrominance of channel a and channel b. The MSE values of regressor model on test dataset was optimal. Colorizer model was implemented with upsampling and downsampling convolution layer. The gray channels of color face images were passed to predict the channel a and channel b, thus colorized image was formed. Using this colorizer model and weight trained, it was again trained on different dataset like NCD. This concept of transfer learning helped retrain on new dataset much faster with existing trained model and colorization was achieved on NCD dataset.

H) Optional Credit: -

A) GPU –

The following experiment was run using benchmark function of torch library. This experiment was tested with 1 thread and 5 epochs. The results shows that GPU is much faster than CPU.

Table 5: GPU v/s CPU speed comparison table

Model	Device	Time (s)
Regressor	GPU (cuda:0)	26.78 s
	CPU	224.98 s
Colorizer	GPU (cuda:0)	60.92 s
	CPU	74.76 s
Transfer Learning	GPU (cuda:0)	10.84 s
	CPU	17.81 s

B) Tanh -

The final_tanh.ipynb contains the implementation of the same code with changes in the activation function. Instead of nn.ReLU(), nn.Tanh() is used. The below figure shows the output for the face image colorization and transfer learning NCD image colorization. From the loss curve for colorization model, we can observe that training with tanh is stable than ReLU as the loss are not fluctuating much for higher epochs.

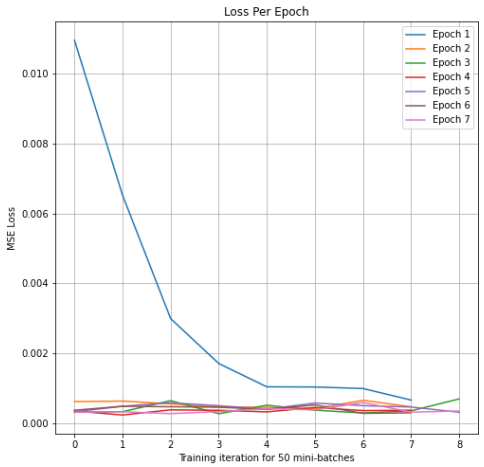

Figure 9: Loss curve for each epoch while training Colorizer with Tanh activations

Figure 10: Results of Colorizer with tanh activation on Test Face Images

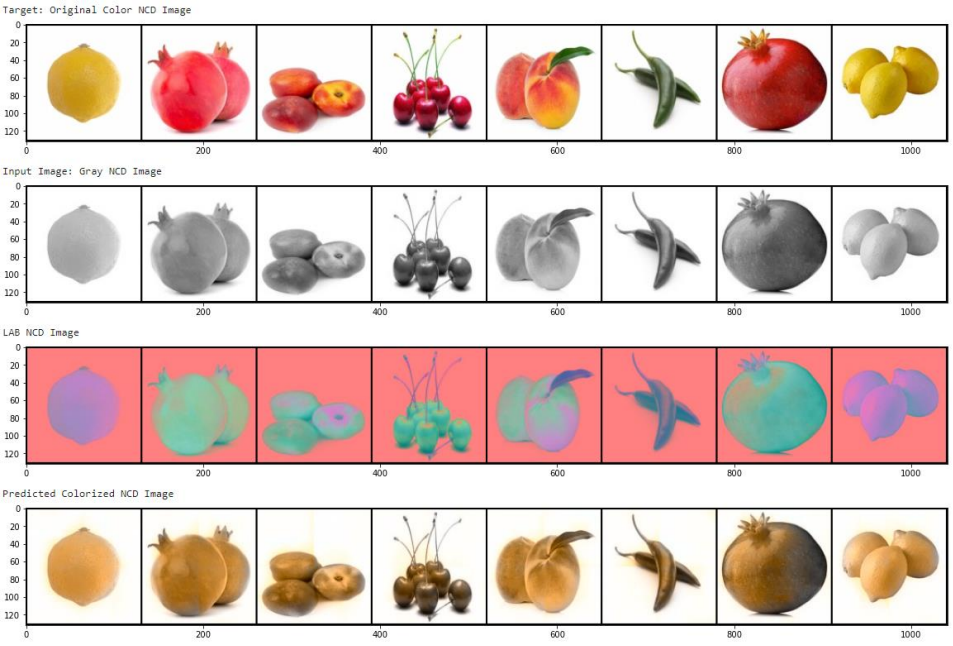

Figure 11: Results of Transfer model with Tanh activation on Test NCD Images

References: -

1) ANWAR S., TAHIR M., LI C., MIAN A., KHAN F. S., MUZAFFAR A. W.: Image colorization: A survey and dataset. arXiv preprint arXiv:2008.10774 (2020).
2) IIZUKA S., SIMO-SERRA E., ISHIKAW H.: Let there be color!: Joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. ACM Transactions on Graphics (Proc. of SIGGRAPH 2016) (2016), 110:1–110:11.
3) https://pytorch.org/vision/stable/transforms.html
4) https://pytorch.org/docs/stable/nn.html
5) https://pytorch.org/tutorials/recipes/recipes/benchmark.html