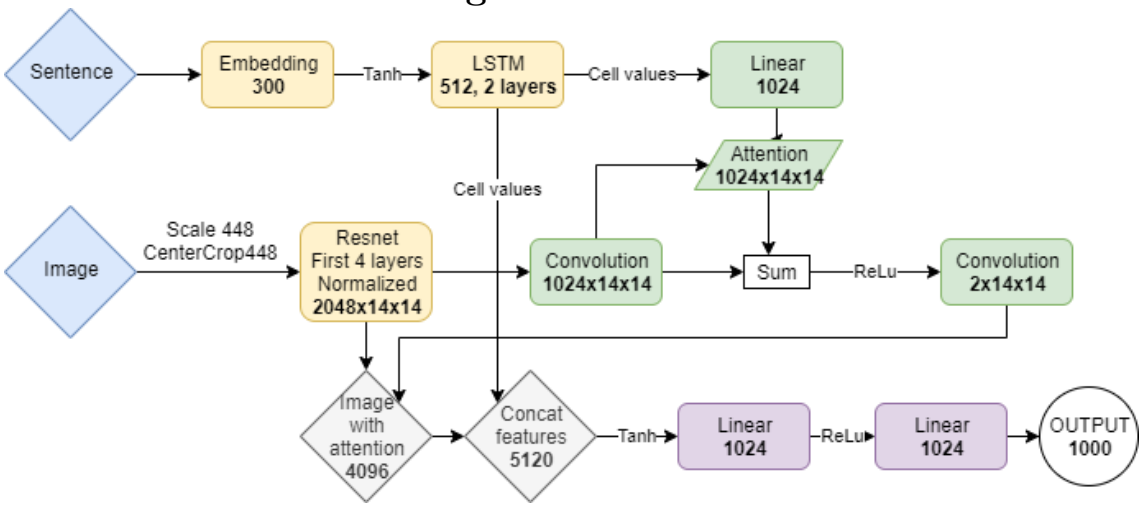
Alexander Chapanin 333815751
Hamza Khatieb 315398933

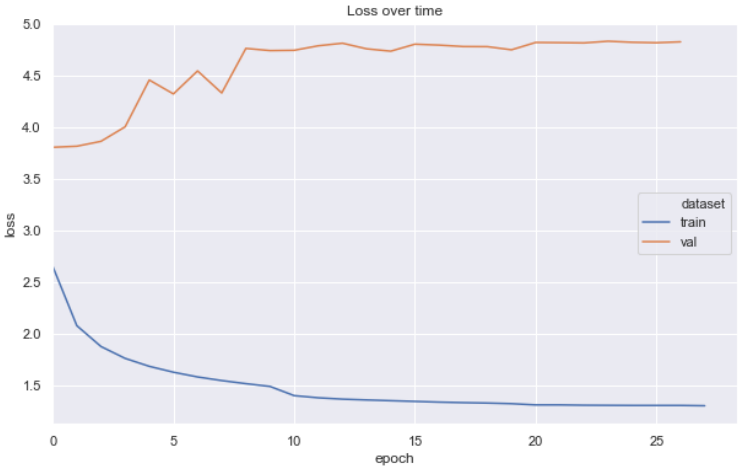
Architecture and training:

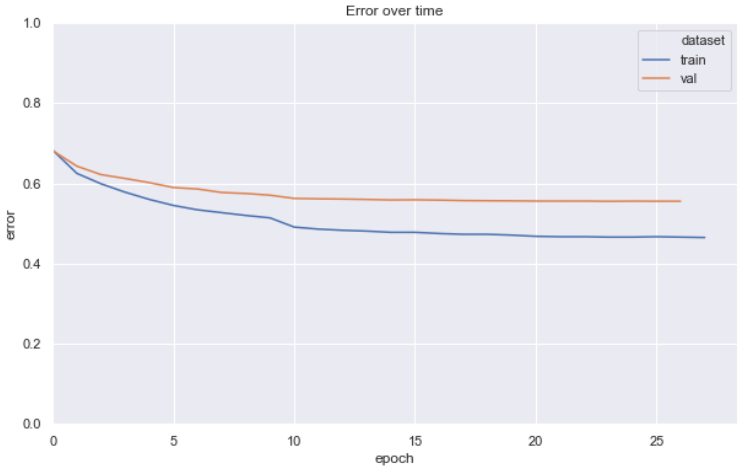


Bold number describes output size of the layer. Yellow layers are preprocessing, green are attention model, purple is combined model.

- We interpreted task as multi-class classification with top-1000 most popular 'multi-choice-winner' answers from train set as label space. Cross Entropy loss was used for training, and only multi choice winner was the target label.
- Learning rate was 0.001 and was updated every 10 epochs with StepLR(gamma 0.1)
- Tanh was used as activation function, as we saw it widely used in open-source solutions.
- Model was initiated with xavier uniform function. Biases were set to 0.
- Attention model created 2 glimpses.
- LSTM's cell values(from both layers, total 1024) were used to encode the whole sentence, output was ignored.
- Dropout was set to 0.5, and used after embedding in question preprocessing, before linear and convolutions in attention model, and before 2 combined linear layers.

Results:





Error above is 0-1 error w.r.t. multi-choice winner. Value of VQA accuracy, considering all other answers(https://visualqa.org/evaluation.html) is higher, but wasn't measured explicitly during train time, because of lack of time.

Wierd difference between train and validation loss is unexpected, but we didn't find out why this happens.

Final 0-1 accuracy(val) of multi-choice winner: 44.45

Final VQA accuracy(val) of all answers: 53.23

Attempts:

During our attempts we focused on testing different architectures instead of tuning the same model.

1. First attempt used embedding+LSTM+linear for text preprocessing and pre-trained vgg19 network without last layer with new linear layer for image preprocessing. After, both vectors were concateneted and passed through 2 additional fully connected linear layers. It was very fast if vgg19 was used before training and it's result was stored aside.This model archieved 38% 0-1 accuracy on validation set, but tend to overfit training data after small amount of epochs.

2. Second attempt was very similar to the first one, but instead of concatenating 2 vectors they were brung to the same length, multiplied and passed on to 2 linear layers. This model archieved 40% 0-1 accuracy on validation set, but had the pros and cons as model above. We also tried to take not only cell values of both LSTM layers, but also hidden states.

3. But we thought there may be the problem with image processing, because we didn't receive any dimensional features, only 4096 values of last vgg19 layer, which may be not holding all required information to answer the question. So instead, inspired by open-source solution we used earlier output of other network: 4th layer of resnet, which returns 2048x14x14 features for each image. Also, we took attention model(and parts of the code) which can fully utilize this information. Full network and it's results were described above. Question processing remained the same without last linear layer.

Conclusions:

It is possible to build relatively simple network, utilitizing pre-trained models impletemented in pytorch. But when using them we should be careful to choose at which layer we take the information. Also, adding full pre-trained model is ineffective, as it greatly increases time for training/evaluation. Better to preprocess images and store aside(although resnet's preprocessing took ˜100gb to store). But this approach also has it's downsides: we can't make randomized image preprocessing(e.g. randomCrop) if we want to save preprocess result

Tuning such a network is not easy task because of time requirements. Dropout is required to avoid overfit in such a hard task.

Preprocessing is important part of workflow, and it should be done carefully, including image preprocessing, as it affect results a lot.

P.S. Our code(both train and eval) assumes same file structure as VQA API. I.e. in current folder there are folders ./Images/train2014, ./Questions, ./Images/val2014/, ./Annotations. We also assume that there is folder /StudentData/ where we can store preprocessed images. Please note if try to reproduce.