# Deep Learning - Project

Răzvan-George Costea

January 2025

## 1 Introduction

In this project, we had to classify using neural networks if image-text pairs are matching or not. The approach is to learn embeddings for both images and captions and then classify if they match or not using a fully connected neural network.

## 2 Data preparation

### 2.1 Text data

Most of the processing is done on text data. Each caption is tokenized, and a vocabulary is built, which is a simple word to index dictionary. The tokenization process includes multiple spaces reduction, punctuation cleaning, and stemming. The tokens are obtained using nltk library. After the vocabulary is built, every caption is encoded and padded to the maximum length which is 32 words. The maximum length was determined by inspecting the data, i.e. finding the maximum number of words of a caption. An encoded caption will be a vector of 32 integers where the first N elements will be the index of the word that was encoded, N meaning the length of the caption. The other 32-N elements will be padding. The vocabulary contains the encoding of "$<mask>$" (used for the MLM pretraining experiment), "$<pad>$", "$<unk>$" tokens.

The only pre-processing done to the images is a resize.

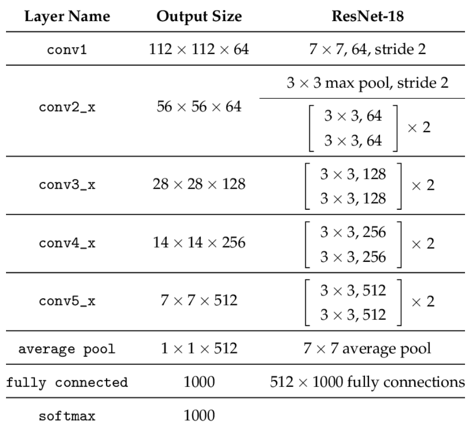
Everything is done in a Dataset class from torch.

## 3 Modeling

For this task the approach is to learn a representation of images and one for captions and classify if they are a match using a fully connected network.

### 3.1 Image model

For the image representation we built two convolutional models, a CNN encoder inspired by ResNet-18 and a custom Autoencoder that was used for



| Layer Name | Output Size | ResNet-18 |
|---|---|---|
| conv1 | $112 \times 112 \times 64$ | $7 \times 7$, 64, stride 2 |
| conv2_x | $56 \times 56 \times 64$ | $3 \times 3$ max pool, stride 2 |
| | | $\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$ |
| conv3_x | $28 \times 28 \times 128$ | $\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$ |
| conv4_x | $14 \times 14 \times 256$ | $\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$ |
| conv5_x | $7 \times 7 \times 512$ | $\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$ |
| average pool | $1 \times 1 \times 512$ | $7 \times 7$ average pool |
| fully connected | 1000 | $512 \times 1000$ fully connections |
| softmax | 1000 | |

Figure 1: ResNet-18

unsupervised pretraining experiment.

The ResNet-18 inspired network has the architecture from Figure 1 and can be seen in detail in the extended material (Figure 7). This network is used in the experiments with random weights or with pretraind weights on a pretext task of predicting the rotation of an image.

The Autoencoder is a simple Encoder-Decoder type of CNN. The use of this network is to learn the representation of the images without the text network training to interfere with it. Therefore, the network's encoder is used with pretrained weight on the reconstruction task. It's architecture can be visualized in the extended material (Figure 8)

## 3.2 Text model

Because text data is sequential we used Recurrent Neural Networks. Because the standard RNNs suffer from vanishing gradients for long sequences we employ Long Short Term Memory architecture. It can also be seen in the extended material Figure **??**. The number of layers is a hyperparameter that was tuned, but from the experiments the best model has one LSTM layer.

The LSTM architecture was used in different experiments both with random weight and pretrained weight. The pretraining was done on Mask Language Modeling task. The architecture is the following:

$LanguageModel($
$(embedding) : Embedding(3977, 256, padding\_idx = 0)$
$(lstm) : LSTM(256, 512, batch\_first = True)$
$)$

The other language model was a simple fully connected network used with random weight:

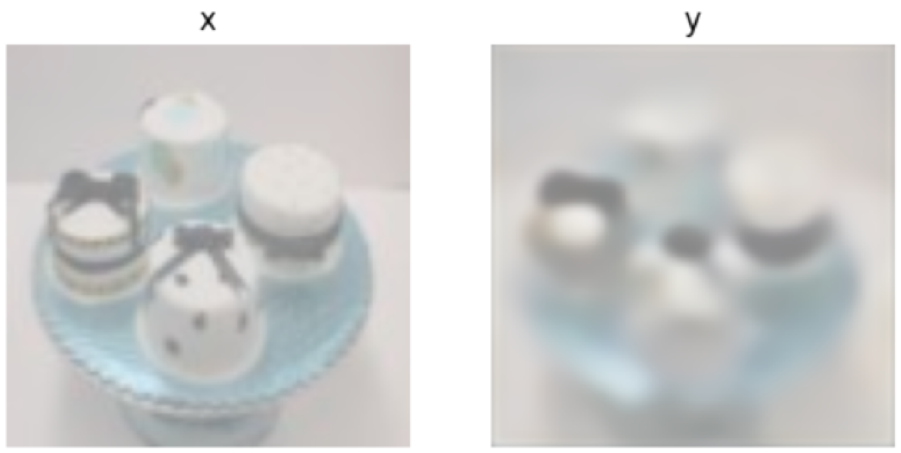


Figure 2: Autoencoder pretraining result

LanguageModel
(embedding): Embedding(3977, 32)
(mlp): Sequential(
(0): Linear(in$_{f}eatures = 1024, out_{f}eatures = 512, bias = True$)
(1): ReLU()
)

## 4 Training and evaluating

### 4.1 Pretraining of image and text models

In the previous section we mentioned that each model, except FC for text, was also used with pretrained weights on the given data.

The CNN was pretrained to predict the rotation of an image. This task is a classification with 5 classes meaning a degree of rotation (-60, -30, 0, 30, 60). The optimizer used is Adam and the loss function is CrossEntropyLoss

The LSTM was trained to predict the masked word in a caption. This is done by using a special token named $< mask >$. A random word is replaced with this token and then the model is trained to predict the missing token. It is treated as a classification problem. The optimizer used is Adam and the loss function is CrossEntropyLoss

The Autoencoder is trained in an unsupervised fashion. The model is trying to reconstruct the input image, thus learning the representation of the it. In the training for the image-text pair matching only the encoder is used. The optimizer used is Adam and the loss function is MAE loss. The results can be visualized in Figure 2

## 4.2 Final models

In the previous section we discussed the individual components of a final model that will be trained. Now we defined how this components are used for the classification task. Each final model has two components, a backbone and a classifier, which is a FC network. The backbone is also composed of two components, a visual model and a langauge model. Therefore the final models are: $(CNN + LSTM) -> FC, (CNN + FC) -> FC, (AE - Encoder + LSTM) -> FC$.

The problem was approached as a classification between two separate classes (2 neurons as output, where each neuron represents the probability of that a sample being in class 0 or class 1) and a binary classification based on a threshold (1 neuron as output, where the output represents the probability that a sample is matching or not).

## 4.3 Experiments

In this competition we have done a number of 46 experiments. All of them have either CrossEntropyLoss, either BinaryCrossEntropy as loss function. Also, all of them all trained with 64 batch size on P100 GPU from kaggle during 50 epochs. In general, each experiment have a validation accuracy around $60\% - 65\%$ on validation data and the best experiments are up to 70% accuracy on validation. The experiments are both with random and pretrained weights with the models listed above. On the test data the models performed worse, therefore we have a massive drop in accuracy, from 70% to around $52\% - 54\%$. The main experiment are the following:

| Id | Optimizer and LR | Loss function | Vision Parameters | Language Parameters | Classifier Parameters |
|---|---|---|---|---|---|
| 1 | Adam 2e-4 | CELoss | CNN | LSTM, 256, 512, 1 | [1024, 512, 256, 2], ReLU |
| 2 | Adam 1e-4 | CELoss | CNN | LSTM, 256, 512, 1 | [1024, 512, 256, 2], ReLU |
| 12 | Adam 2e-4 | CELoss | CNN | LSTM, 256, 512, 2 | [1024, 512, 256, 2], ReLU |
| 13 | Adam 2e-4 | CELoss | CNN | LSTM, 256, 512, 2 | [1024, 64, 32, 2], ReLU |
| 17 | Adam 2e-4 | CELoss | PT CNN | LSTM, 256, 512, 1 | [1024, 512, 256, 2], ReLU |
| 21 | AdamW 2e-4 | CELoss | PT CNN | LSTM, 256, 512, 2 | [1024, 512, 256, 2], ReLU |
| 24 | Adam 2e-4 | CELoss | PT CNN | LSTM, 256, 512, 3 | [1024, 512, 256, 2], ReLU |
| 25 | Adam 2e-4 | CELoss | PT CNN | LSTM, 256, 512, 1 | [1024, 2], ReLU |
| 28 | Adam 2e-4 | CELoss | PT CNN | PT LSTM, 256, 512, 1 | [1024, 2], ReLU |
| 30 | Adam 2e-4 | CELoss | PT CNN | MLP, 32, [1024, 512], ReLU | [1024, 512, 256, 2], ReLU |
| 31 | Adam 2e-4 | CELoss | CNN | MLP, 32, [1024, 512], ReLU | [1024, 512, 256, 2], ReLU |
| 32 | Adam 1e-3 | CELoss | PT CNN | MLP, 32, [1024, 512], ReLU | [1024, 512, 256, 2], ReLU |
| 34 | Adam 1e-4 | CELoss | PT CNN | MLP, 32, [1024, 512], ReLU | [1024, 512, 256, 2], ReLU |
| 35 | Adam 2e-4 | CELoss | PT AE | LSTM 256, 512, 1 | [1088, 512, 256, 2], ReLU |
| 36 | Adam 2e-4 | CELoss | PT AE | MLP, 32, [1024, 512], ReLU | [1088, 512, 256, 2], ReLU |
| 37 | Adam 2e-4 | BCELoss, 0.5 | CNN | LSTM, 256, 512, 1 | [1024, 512, 256, 1], Tanh |
| 38 | Adam 2e-4 | BCELoss, 0.5 | CNN | LSTM, 256, 512, 1 | [1024, 512, 256, 1], ReLU |
| 42 | Adam 2e-4 | BCELoss, 0.9 | CNN | LSTM, 256, 512, 1 | [1024, 512, 256, 1], Tanh |
| 44 | Adam 2e-4 | BCELoss, 0.3 | CNN | LSTM, 256, 512, 1 | [1024, 512, 256, 1], Tanh |

## 5 Results

### 5.1 Main experiments

The following table shows the results for the main experiments listed above. The top 2 submissions selected for the competition are experiments 1 and 2. Experiment 17 had the best result in Private testing, but it was not selected for submission due to lower accuracy on Public testing

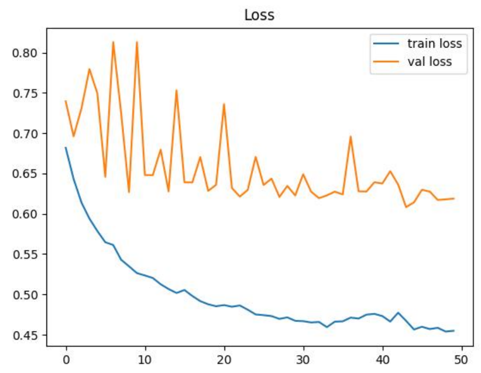
| Id | Validation accuracy | Private test accuracy | Public test accuracy |
|---|---|---|---|
| 1 | 0.69933 | 0.51625 | **0.51750** |
| 2 | 0.69833 | **0.53625** | **0.51500** |
| 12 | 0.62166 | 0.49250 | 0.47250 |
| 13 | 0.70033 | 0.50875 | 0.48750 |
| 17 | 0.69166 | **0.54375** | 0.51000 |
| 21 | 0.51866 | 0.50812 | 0.51250 |
| 24 | 0.69600 | 0.50312 | 0.47000 |
| 25 | 0.69533 | 0.50687 | 0.48500 |
| 28 | 0.64433 | 0.51187 | 0.46750 |
| 30 | 0.67433 | 0.48500 | 0.49500 |
| 31 | 0.67133 | 0.51687 | 0.49500 |
| 32 | 0.61633 | 0.49750 | 0.49000 |
| 34 | 0.66266 | 0.52062 | 0.47000 |
| 35 | 0.63300 | 0.51250 | 0.49500 |
| 36 | 0.62166 | 0.49500 | 0.47500 |
| 37 | 0.67666 | 0.49187 | 0.49500 |
| 38 | 0.69166 | 0.48437 | 0.47750 |
| 42 | 0.68000 | 0.50812 | 0.47000 |
| 44 | 0.67600 | 0.50812 | 0.47000 |

Plots from Figure 3 shows the accuracy and loss for train and validation datasets over epochs for the experiments 1, 2 and 17.

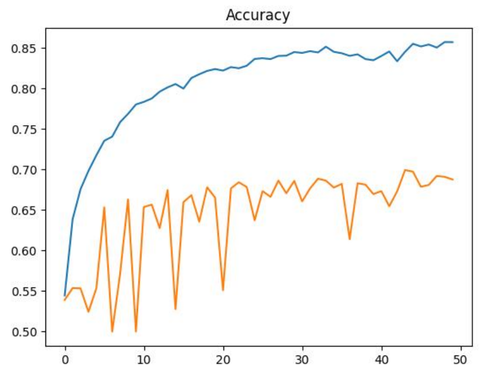
Confusion matrix are displayed in Figures 4, 5, 6

## 6 Extended Material

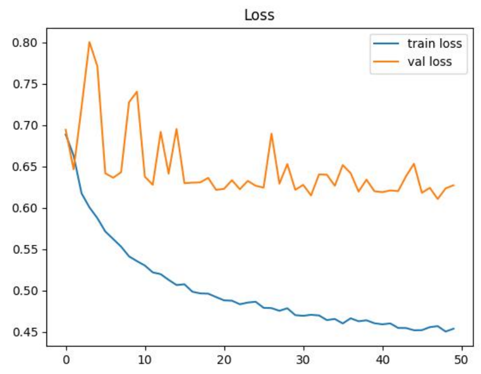
### 6.1 Architectures



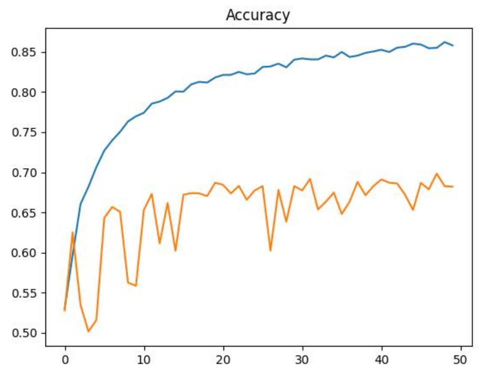
(a) Experiment 1 loss



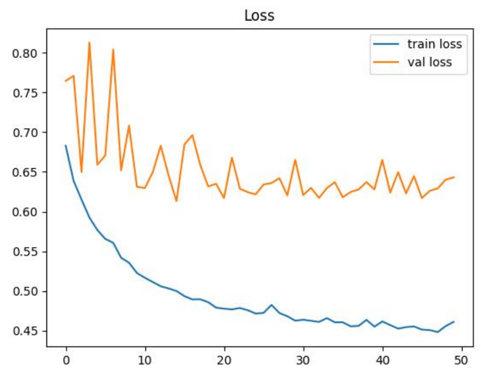
(b) Experiment 1 accuracy



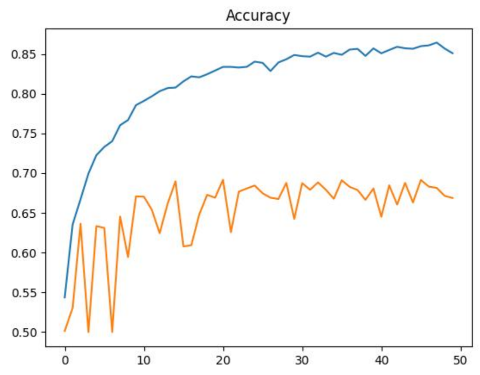
(c) Experiment 2 loss



(d) Experiment 2 accuracy



(e) Experiment 17 loss



(f) Experiment 17 accuracy

Figure 3: Plots with train and validation loss and accuracy

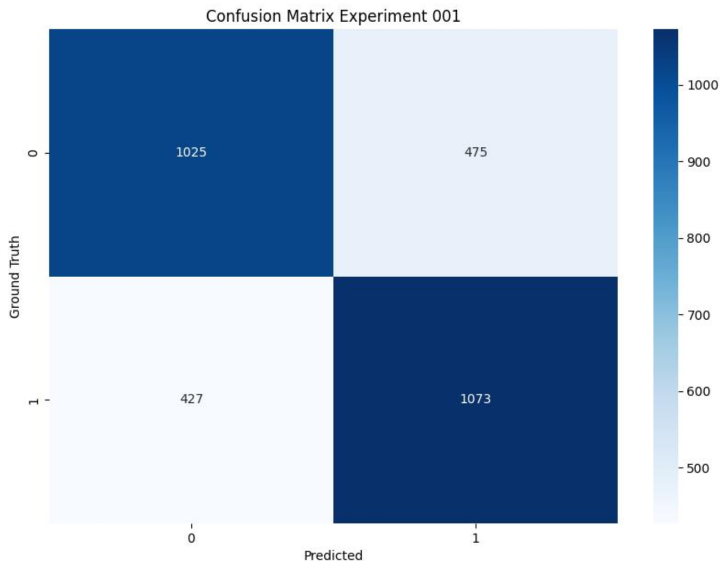


Figure 4: Confusion matrix for experiment 1

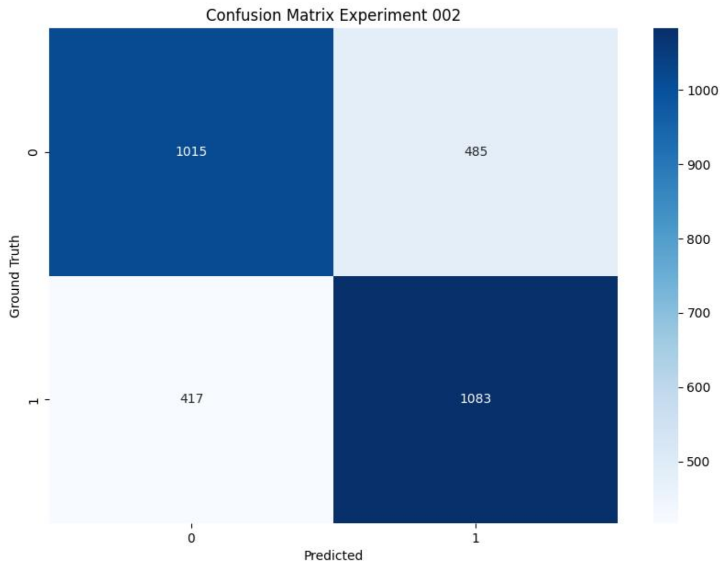


Figure 5: Confusion matrix for experiment 2

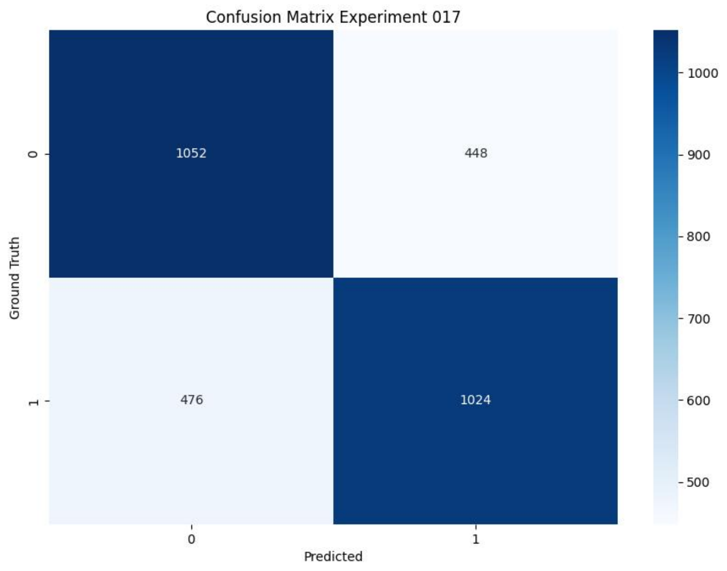


Figure 6: Confusion matrix for experiment 17

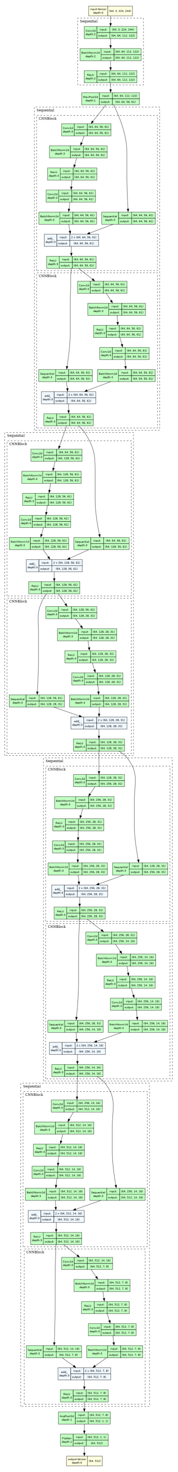


Figure 7: CNN model inspired by ResNet-18

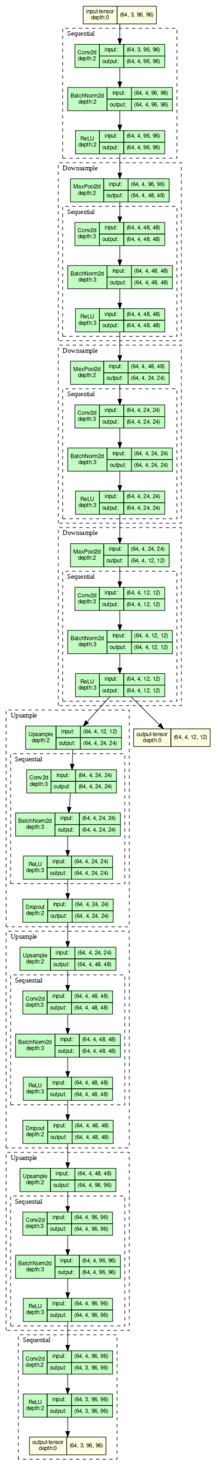


Figure 8: Autoencoder model