RESULTS

PREPROCESSING

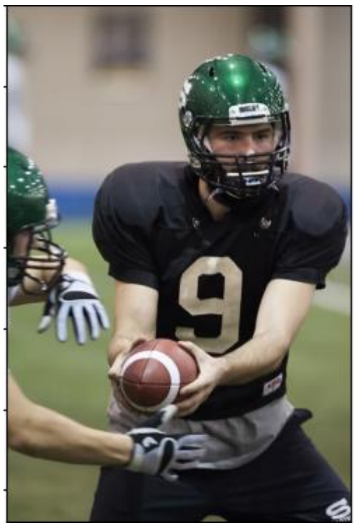
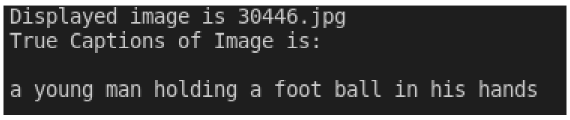


Figure 1: Sample image and caption



```
Displayed image is 30446.jpg
True Captions of Image is:

a young man holding a foot ball in his hands
```

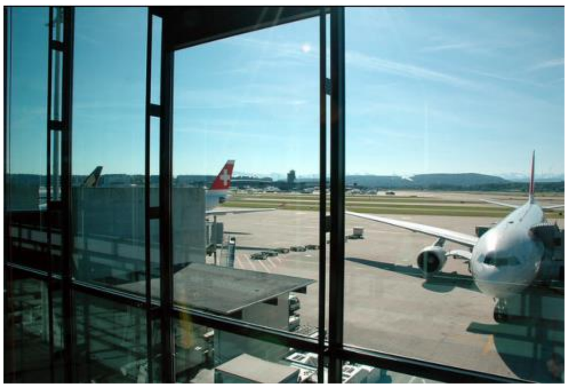
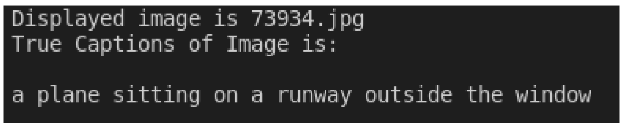


Figure 2: Sample image and caption



```
Displayed image is 73934.jpg
True Captions of Image is:

a plane sitting on a runway outside the window
```

The vocabulary size of the whole dataset is 1004 and the max caption length is 17. The captions start with < START > signal and when the caption ends, if there were remaining slots, the ending signal < END > fills those spaces.

I split the training data as %85 of the data for training and %15 of the data for validation. After training and optimizing the data, a whole another test set it used for testing the accuracy of the model and relevance of the captions.

For the preprocessing part, the main problem was that images had different sizes. Since, I was using transfer learning in the CNN part, the popular CNN models that I used like ResNet, VGGNet, all accept the images as input with a size of 224x224x3 where 3 denotes RGB. So that, I had to convert those images to meet the model requirements. In the conversion process, the transforms module of Pytorch library is used.

Also, I did not directly convert the images to the desired sizes. The images were taken into a process called Data Augmentations which is a widely used technique to increase the sample size by adding slight changes into the data and add as another input to the input pool. But in my case, since I have 4 or 5 copies for each image, I did not want to train the model 5 times with the same image so I add augmentations to it with a probability of 0.5. This process will act as a regularizer and helps to reduce overfitting. The Data Augmentations I used are Random Resized Crop and Random Horizontal Flip. The first one randomly crops a part of the image and then resizes it to the desired size and the second one randomly flips respect to the horizontal axis. These both have 0.5 probability to occur.

Finally, as a part of preprocessing, the data is normalized for each of the RGB channels separately.

CNN ENCODING AND TRANFER LEARNING

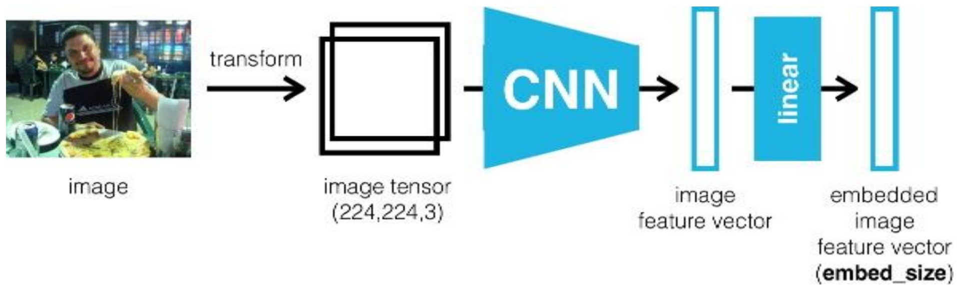


Figure 3: Encoder CNN

As it is mentioned before, the first part of the process is to extract features from the input images by using CNN in order to fed them into the RNN. Training a network for this process with a dataset like this would be highly costly and it will take too much time. Furthermore, the trained set will not give good results in our situation. So, a widely used method for this problem is the transfer learning method which I also used in my design. With this method, I can use a pretrained model and make some changes to fit the model in our design. The general idea of using transfer learning can be summed in 5 steps:

- Load in a pre-trained CNN model trained on a large dataset
- Freeze parameters (weights) in model's lower convolutional layers
- Add custom classifier with several layers of trainable parameters to model
- Train classifier layers on training data available for task
- Fine-tune hyperparameters and unfreeze more layers as needed [3]

I used ResNet18, VGGNet19 and V3 Inception for the pretrained model and also add another fully connected layer at the end of the models rather than the SoftMax layer. This is added for two purposes. The first purpose is to make the final output size desired so that I can connect the CNN and RNN models. The second purpose is that to train the fully connected layer with the error coming from the RNN part of the model. In transfer learning, generally two different approaches are widely used. The first one is to only train the last fully connected layer and make the output size desired. And the second one is to train the all fully connected layers at the end of the model without adding a new layer. For simplicity and less cost and time, I choose to go one with the first approach. The ResNet architecture and the Tensorboard visualization of ResNet and V3 Inception models are shown below.

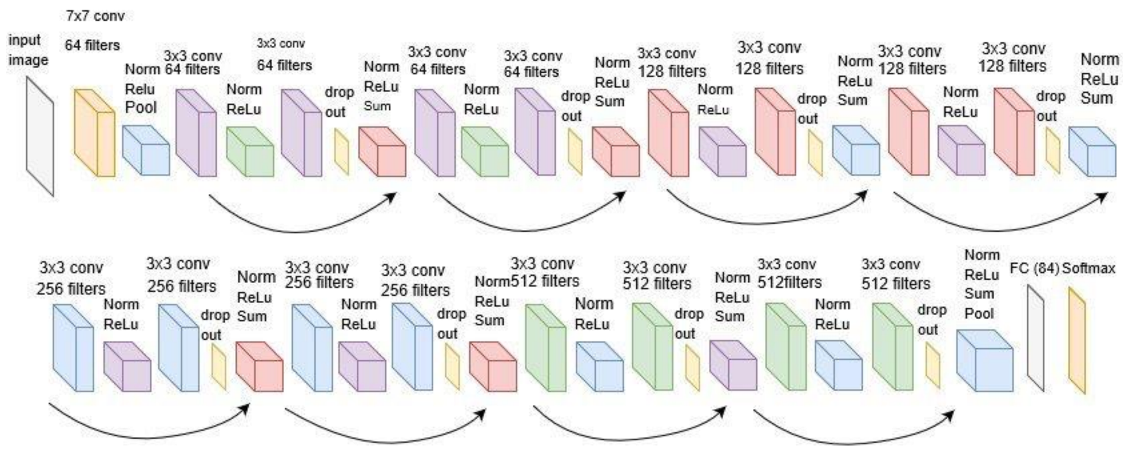


Figure 4: ResNet 18 architecture

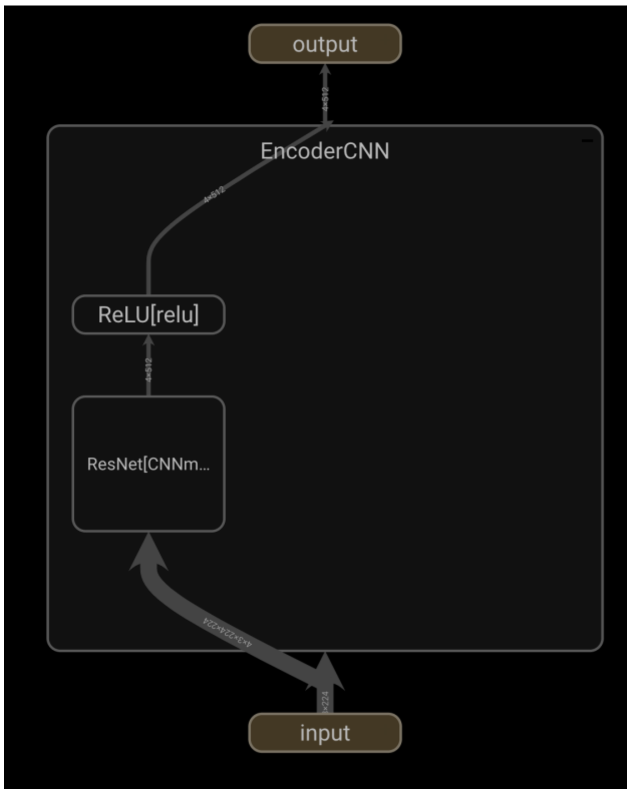


Figure 5: ResNet18 model with FC in Tensorboard

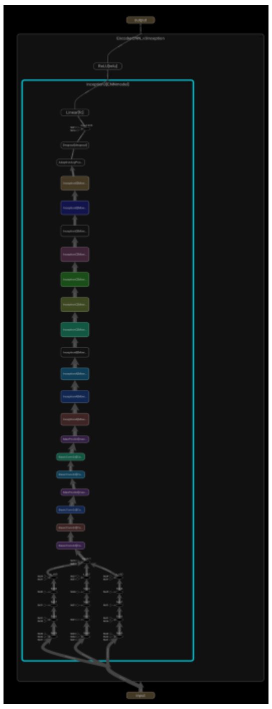


Figure 6: V3 Inception model inside layers in Tensorboard

DECODER WITH TEACHER FORCING

Teacher forcing is a widely used technique in training Recurrent Neural Networks (RNN). In Natural Language Processing (like image captioning, machine translation), recurrent neural network mechanism is used and in order to perform RNN, teacher forcing methodology is generally used for training process. In this technique, I have a ground truth of captions for each image and in training process, I feed RNN with these ground truths instead of the previous state's output. The methodology can be explained more clearly with this example. I have a ground truth sequence which is "A man is playing basketball ". My designed model made a mistake at third word. Model creates a starting sentence as "A man are …" respectively. In teacher forcing methodology, I am feeding the model's forth word assumption with "is" instead of "are". The previous cell state feeds this layer as previous layer's memory also [3]. The overall look of Teacher Forcing method is as following figure:

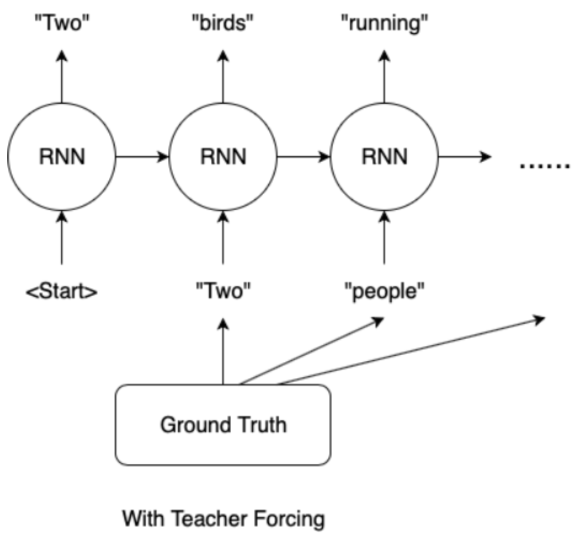


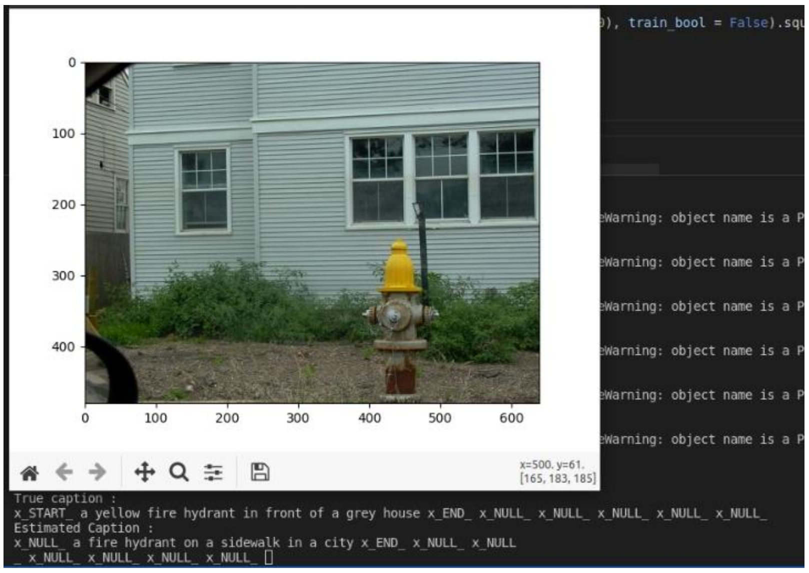
Figure 7: RNN Algorithm with Teacher Forcing Methodology [3]

Pros:
Training with Teacher Forcing converges faster than other methods [3]. If I do not use teacher forcing, the hidden states of model will be updated by wrong predictions from previous time step [3]. Therefore, errors will be accumulated through time steps.
Cons:
In the test part, the model has not captioned as ground truths as if model does not know the real captions. Therefore, in inference part, model has no ground truth which cause different methodology of testing test data from training case. The model needs to feed next time step with previous step's prediction which means there is a discrepancy between training and testing part [3].

Due to above mentioned reasons, I modeled my RNN by using teacher forcing methodology.

RESULTS

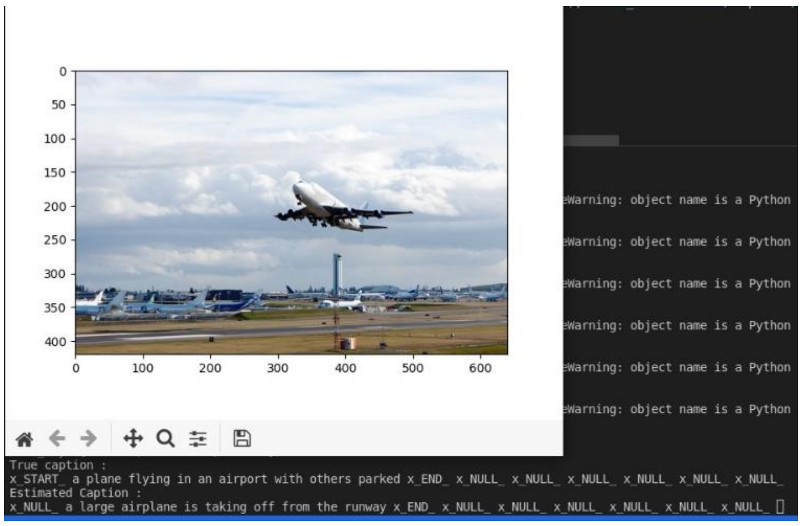
Some results with true captions and estimated captions are shown below.



```
), train_bool = False).squ
eWarning: object name is a P
eWarning: object name is a P
eWarning: object name is a P
eWarning: object name is a P
eWarning: object name is a P
eWarning: object name is a P
True caption :
x_START_ a yellow fire hydrant in front of a grey house x_END_ x_NULL_ x_NULL_ x_NULL_ x_NULL_ x_NULL_
Estimated Caption :
x_NULL_ a fire hydrant on a sidewalk in a city x_END_ x_NULL_ x_NULL
_ x_NULL_ x_NULL_ x_NULL_ x_NULL_ ▯
```

Figure :Sample image with estimation and true label

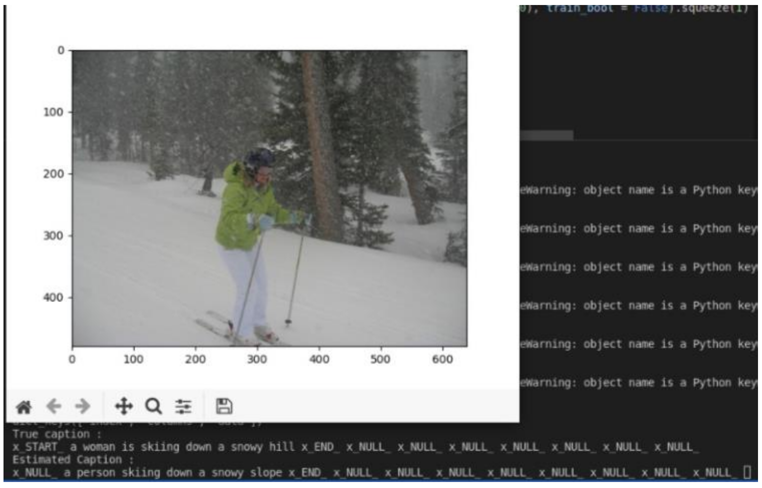
True caption: A yellow fire hydrant in front of grey house
Estimated Caption: A fire hydrant on sidewalk in a city



```
eWarning: object name is a Python
eWarning: object name is a Python
eWarning: object name is a Python
eWarning: object name is a Python
eWarning: object name is a Python
eWarning: object name is a Python
True caption :
x_START_ a plane flying in an airport with others parked x_END_ x_NULL_ x_NULL_ x_NULL_ x_NULL_ x_NULL_ x_NULL_
Estimated Caption :
x_NULL_ a large airplane is taking off from the runway x_END_ x_NULL_ x_NULL_ x_NULL_ x_NULL_ x_NULL_ x_NULL_ ▯
```

Figure :Sample image with estimation and true label

True caption: A plane flying in an airport with others parked
Estimated Caption: A large airplane is taking off from the runway



```
0), train_bool = False).squeeze(1)

eWarning: object name is a Python key

eWarning: object name is a Python key

eWarning: object name is a Python key

eWarning: object name is a Python key

eWarning: object name is a Python key

eWarning: object name is a Python key

True caption :
x_START_ a woman is skiing down a snowy hill x_END_ x_NULL_ x_NULL_ x_NULL_ x_NULL_ x_NULL_ x_NULL_ x_NULL_
Estimated Caption :
x_NULL_ a person skiing down a snowy slope x_END_ x_NULL_ x_NULL_ x_NULL_ x_NULL_ x_NULL_ x_NULL_ x_NULL_ x_NULL_
```

Figure :Sample image with estimation and true label

True caption: A woman is skiing down a snowy hill
Estimated Caption: A person skiing down a snowy slope