CS5242: Neural Networks and Deep Learning
Video Captioning

Rohit Jain
E0674484
e0674484@u.nus.edu

Archit Aggarwal
E0676680
e0676680@u.nus.edu

Brian K Chen
E0694208
e0694208@u.nus.edu

Abstract - The natural human ability to perceive a sequence of events to determine the objects involved in the sequence and the relation between them is an apt problem for deep neural networks. In this work we undertake this problem by processing a video sequence and relevant caption data to determine the objects and their relationship through deep neural networks. We use the implementation of recent state of the art pre trained CNNs for extracting video frame features and multiple Recurrent Neural Networks for handling sequences of data. We approach the problem with trying out multiple architectures involving deep convolutional neural networks and recurrent neural networks. On our best proposed model, we are able to get very decent object-relationship pairs which lands a rank in the top 5 on the Kaggle leaderboard.

1 Introduction

The most common cognitive ability granted to humans is the ability to process a sequence of events to make determinations in terms of associations and relationships and to produce some conclusion. Granting machines this ability to make informed decisions on unseen data is now possible due to recent advances in computer science and AI. The problem at hand focuses on a similar task where we aim to teach a system to be able to determine objects in a sequence of events and draw some relation between these objects from a predetermined set of conclusions, somewhat replicating the same ability of living beings.

The data or the sequence of events here refers to videos which have been broken down into frames, processed in chronological order. We discovered that the problem at hand needs to be approached as a set of two individual problems, first being to extract features out of the image which will contain information of objects and its relationship, and the second part where we unroll the caption taking combined features of all 30 frames into account. For the task of feature extraction, we implement and modify some of the recent state of the art models like EfficientNet, Resnet 152, InceptionNetV4, pretrained on ImageNet dataset. For sequencing features for 30 frames and for unrolling the caption (object-relationship triplets) RNN's were the most suitable choice given their proven efficacy in sequence to sequence modelling.

The rest of this script is organized as follows, Section 2 presents a concise theoretical background, Section 3 describes the proposed approach, Section 4 describes our experimental setup, comparison and results and we conclude in Section 5.

2 Background Information

2.1 Feature Extraction

Feature extraction is a form of dimensionality reduction to extract key information from the video frames while removing the redundant information. Using all the given pixels without convolution and pooling will lead to 30 x 1280 x 720 (~ 30 million) parameters otherwise, for each of the videos. CNNs allow us to extract abstract features from raw image pixels, shallow convolutional layers learn features like-edges, corners, ridges etc., while deeper convolutional layers extract complex textures and patterns like objects or part of the objects. Since the object classes for images can all be found on ImageNet data, we explored several popular convolutional neural nets pre-trained on ImageNet- EfficientNet-B7, Resnet 152, InceptionNetV4 for feature extraction. Although the field of ConvNets is progressing at a break-neck pace, all of these models can still be considered state of the art, especially EfficientNet.

ResNet is a model introduced in 2015 which uses an identity shortcut connection to skip multiple layers. This prevents gradient vanishing during backpropagation and allows the model to include a much larger number of layers as compared to previous models such as *AlexNet* and *VGG*. *ResNet128* has a top-5 error of 3.57%. *InceptionNets* are a type of

network architecture which utilizes multiple inception blocks. These blocks are an ensemble of separate convolutional layers each with different characteristics. This allows us to extract different kinds of features. *InceptionNet V4* also utilizes batch normalization and introduces factorization in the layers to reduce dimensionality and hence computational cost. Compared to other versions of *InceptionNet*, *InceptionNet v4* contains more inception modules and has proven to be very effective. InceptionNet V4 has a top-5 error of 3.08%. *EfficientNets* are truly state of the art, only introduced in late 2019. Essentially it builds upon previous Convolutional Neural networks by adopting a principled method of scaling up networks by uniformly scaling all dimensions using a simple compound coefficient. The one problem which eventually prevented us from using this network is that, while computationally it is very efficient, compared to other scaled networks, the features extracted require a large amount of memory, which we did not have access to. *EfficientNet B7* has a top-5 error of 2.9%.

2.2 Image Captioning

The second part of the problem is to caption the video (or object-relationship pairs). While Deep Neural Networks are effective given large training sets, they are unable to map sequences to sequences. In the past decade, research has shown that sequence to sequence methods have proven to be very effective. Gated RNNs use gated units such as LSTMs and, more recently, GRUs, to control how the past and present memory is used to update current activation.

LSTMs use an input, output and forget gate within each unit to regulate the flow of information in and out each cell. By doing so, we both increase model capacity while preventing vanishing gradients which can arise when increasing the number of cells within each network. Compared to LSTMs, GRUs are a more simple and modern approach. GRU units possess an update and reset gate. Very similar to an LSTM if you remove the output gate. Because of its simplicity GRUs are less computationally costly as compared to LSTMs and current literature shows it often matches, or at times does better, than LSTM.

To implement these gated units, we consider following different structures for both the encoder and decoder RNNs separately- Single pass RNN, Stacked RNNs, Bidirectional RNNs and Stacked Bidirectional RNNs. RNNs stand for recurrent neural networks and are a class of neural networks in which the nodes for a directed graph. This exploits the temporal features of a sequence and hence it is perfect for encoding and decoding sequences. Bidirectional RNNs are built upon regular RNNs except includes a layer which runs the sequence in the opposite direction. This allows us to exploit both the forward and backwards temporal dependencies of spatio-temporal data which we are working with.

In this work, we consider both stacked RNNs and stacked bidirectional RNNs. This involves adding layers and hence depth to our neural network, thus adding model capacity at the cost of possibly vanishing/exploding gradients and overfitting.

3 Proposed Approach

3.1 Dataset Description and Preparation

The Kaggle dataset comprises a total of 566 videos (447 training and 119 test videos), where each video is split into 30 frames each. The total number of objects are 35 and the total relationships are 82. Given the small size of training data. We used the entire 447 videos for training and did not split it further into validation as it led to reduced accuracy. The training data gave two separate vocabulary files from which the object and the relation needs to be drawn. We chose to combine them into a single vocabulary file and separate objects from relations during inference. A special token <BOS> Beginning of the sequence, is added to the vocabulary which goes as the initial input to the decoder RNN. Image 1 in Annex shows the model.

3.2 Feature Extraction

We employ a separate script tasked for off the shelf feature extraction. The reason for using a separate script is the large time required for the feature extraction (approx 4 hours), thus combining it with model training results in longer execution time. For each model the last classification layer is removed, and average pooling is added to the layer before for extracting the features in the form of an output tensor.

The feature sizes for various models are as follows -

1. Resnet128, output feature tensor size for each video is (30, 2048).
2. InceptionNetV4, output feature tensor size for each video is (30, 1536).

3. EfficientNet B7, output feature tensor size for each video is (30, 2560).

In order to supplement feature extraction, we experimented with a number of data augmentation techniques. In the final setup we used a random resized crop, random rotation, image normalization (using mean and standard deviation), convert image from BGR to RGB and normalize image range from [0, 255] to [0, 1]. During experimental setups it was concluded that adding extreme rotation or flips to the input data lead to the model learning opposed relations so the produced predictions were a mirror of the correct ones. For instance, to-the-left-of was perceived as to-the-right-of. This result was consistently noted across many predictions during manual evaluation and hence we decided to discard high angle rotation and flipping.

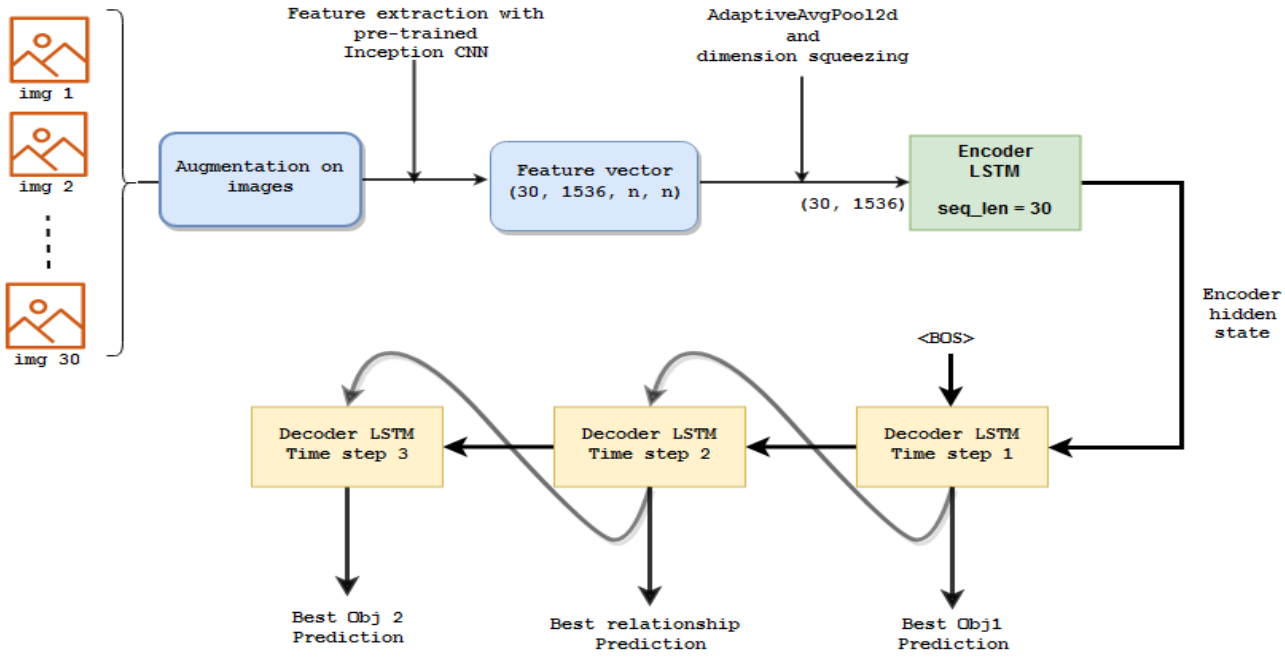

Image 1. The model architecture

3.3 Encoder

The extracted features embeddings are fed to encoder RNN as input which then sequences it. In our approach we decided to go with an LSTM due to its proven applicability in image captioning and seq2seq models. We experimented with stacked LSTMs upto 3 layers, switching the LSTM to a GRU, Bidirectional LSTMs and stacked Bidirectional LSTMs. In recent trends and literature 'deeper' models with more features tend to present better results as seen in feature extraction, but for the RNNs given low complexity of the task, single layer LSTM outperforms all other configurations. Also, in the experimental setting a median size embedding outperforms smaller and larger embeddings. The output hidden state of encoder LSTM, which contains the combined information from all time steps, is fed to the decoder LSTM as initial hidden state.

3.4 Decoder

The Decoder RNN is responsible for producing the final output predictions. The proposed approach has three timesteps. The first time step is fed the <BOS> token and the remaining two timesteps utilize the best prediction from the previous time steps as input. Our model looks up at the vocabulary to predict the best choice at every time step.

4 Experiments

We describe the broad level idea and structure of the model in Section 3. This section aims to provide finer details of the model and present the results with some observations. All the experiments conducted were processed using Google Colab platform and the default GPU provided by Colab.

4.1 Feature Extraction Models

As mentioned earlier, we experimented with EfficientNetB7, ResNet152, InceptionNetV4 models for feature extraction and different hyperparameters and variations in RNNs, which led to top 5 prediction accuracies as depicted in Table 1.

Table 1: Few tried out configurations and results

Feature extractor CNN	Encoder RNN	Decoder RNN	Top 5 score
ResNet128	Single layered LSTM	Single layered LSTM	0.723
ResNet128	Stacked (Multi-layer) LSTM	Stacked (Multi-layer) LSTM	0.712
ResNet128	Bidirectional LSTM	Bidirectional LSTM	0.707
ResNet128	Single layered GRU	Single layered GRU	0.719
EfficientNetB7	Single layered LSTM	Single layered LSTM	0.733
EfficientNetB7	Single layered GRU	Single layered GRU	0.728
EfficientNetB7	Bidirectional LSTM	Single layered LSTM	0.731
InceptionNetV4	**Single layered LSTM**	**Single layered LSTM**	**0.744**
InceptionNetV4	Bidirectional LSTM	Single layered LSTM	0.727
InceptionNetV4	Single layered GRU	Single layered GRU	0.735
InceptionNetV4	**Ensemble** Single layered LSTM Single layered GRU Bidirectional LSTM	**Ensemble** Single layered LSTM Single layered GRU Bidirectional LSTM	0.742

4.3 The final model, best configuration and hyperparameters

4.3.1 Feature Extraction

Table 2 describes all the values used for the final feature extraction via InceptionNetV4

Table 2. Hyperparameters and Data Augmentation for Feature Extraction

Random Crop	Random Resize	Random Rotation	Random Horizontal Flip
400	299	8 Degrees	FALSE

4.3.2 Encoder and Decoder

The range in Table 3 shows the lowest to highest value which was used during experimentation. Due to the large number of models, it is not feasible to mention each value and thus the overall range is shown.

Table 3a .Final Model setup and hyperparameters with complete range of experimental values

Hyperparameter	Learning Rate	Dropout	No. of Stacked Layers	Initialization
Best value	3e-4	0.5	1	Xavier
Range	1e-2 to 5e-4	0 to 0.6	3	He, Xavier, Normalized

Table 3b .Final Model setup and hyperparameters with complete range of experimental values continued

Epochs	Activation Function	Optimizer	Hidden Size	Embedding Size	Top 5 Score
25	RELU	Adam	256	256	0.7445
5 to 50	ReLU, Leaky ReLU, tanh	Adam, RMSProp, AdamW, SGD	128-512	128-512	-

5 Conclusion and Future Work

Given the size of the training dataset, our best performing model gives a competitive result for the predictions of object1-relationship-object2 triplet. The sequence to sequence model seems to perform well for this given task. Key to accuracy in such models is trying out different configurations and hyperparameters tuning, which we explored through the course of this project.

Future work on this project can be collecting more data in terms of quantity and diversity. We noticed the given dataset is slightly skewed with many labels, where a person is riding a bicycle or a dog is with a person. On the other hand, very few labels have objects like cattle, airplane, watercraft.

During inference time, one can adopt beam search which, although is computationally expensive, may yield better results when compared to the greedy approach.

6 Contribution

Table 4 outlines the contribution of the team members for the tasks involved in this project.

Table 4. Contributions of Team Members

Tasks	**Contributors**
Code for feature extraction	Rohit
Model Architecture	Rohit
Pytorch Code Implementation	Rohit
Data Augmentation	Rohit, Archit, Brian
Variations in Architecture	Rohit, Archit, Brian
Hyperparameter tuning	Rohit, Archit, Brian
Final Report	Rohit, Archit, Brian

7 References

1. Venugopalan, S., Rohrbach, M., Donahue, J., Mooney, R., Darrell, T., and Saenko, K. Sequence to sequence-video to text. In Proceedings of the IEEE international conference on computer vision (2015)
2. Feng, V., 2017. Medium. [online] Medium. Available at:<https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035>
3. EfficientNet, Github. [online]. Available at:<https://github.com/lukemelas/EfficientNet-PyTorch>
4. J. Carreira and A. Zisserman, "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 4724-4733, doi: 10.1109/CVPR.2017.502.
5. Tan, M., & Le, Q.V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. ArXiv, abs/1905.11946.
6. Zhiyong Cui, Ruimin Ke, Ziyuan Pu, Yinhai Wang, Stacked bidirectional and unidirectional LSTM recurrent neural network for forecasting network-wide traffic state with missing values, Transportation Research Part C: Emerging Technologies,Volume 118,2020, 102674, ISSN 0968-090X.