Comparing Attention-based Neural Architectures for Video Captioning

Jason Li Helen Qiu

jasonkli@stanford.edu shiqiu21@stanford.edu

Abstract

Integrating visual content understanding with natural language processing to generate captions for videos has been a challenging and critical task in machine learning. The task fundamentally requires an understanding of sequential visual data and also the capacity to translate that understanding into natural language. A key feature in many approaches to this task is attention. Previous work in video captioning largely focuses on temporal attention, where attention is computed on features over time. In this work, we explored attention over space and time with features extracted from Pseudo-3D Residual Net (P3D ResNet), a widely used architecture in activity recognition and compared to performance using temporal attention over 2D-CNN features. Furthermore, we also investigate the use of LSTMs vs. Transformers in the video captioning task. Overall, we did not find that spatio-temporal attention over P3D features improved performance. However, we did find that Transformers outperformed their LSTM counterparts.

1 Introduction

Video captioning is the task of understanding the visual content of video sequence and translating the understanding to an appropriate caption. With video being an important source for human beings to acquire knowledge and communicate emotions, automatic video description generation can have many practical applications in daily scenarios such as video retrieval [1], blind navigation [2], and automatic video subtitling [3]. For example, 300 hours of video are uploaded to YouTube every minute [6], and thus an effective video captioning model has the potential to improve tagging, indexing, and search quality for online videos; in conjunction with speech synthesis technology, video captioning can help produce annotations for the visually impaired people to better engage with the video content [4]. While significant progress has been made in video classification and image captioning using deep Convolutional Neural Network (CNN) [7], the task of video captioning is more difficult and still remains challenging–generating a sentence to characterize a 4- to 10-second-long video clip that consists of 100-250 frames fundamentally requires an understanding of sequential visual data and also the capacity to translate that understanding into natural language. In addition, unlike a video classification task, video description generation requires decoding the semantic content represented by the salient features in the video and converting the information back to language.

Previous work in video captioning has proposed the use of 2D-Convolutional Neural Networks (CNN) for video feature extraction and attention-based Long Short-Term Memory (LSTM) models to capture the most relevant temporal segments in the video for description generation [3]. However, this approach does not include spatial attention over the frame sequence in a video, and only incorporates attention over time by attending to the hidden states of the LSTM. With recent research on Pseudo-3D Residual Net (P3D ResNet), a less computationally expensive 3D-CNN that can learn both spatial and temporal representations of videos [5], we explore whether spatio-temporal representation of attention with P3D can improve the video captioning performance. Here, we explore the use of P3D

in video captioning relative to performance with a traditional 2D-CNN. We also explore the use of LSTMs vs. Transformers to decode the feature representations from P3D or a 2D-CNN.

2 Related Work

Early work on video captioning used metadata to tag videos [8] and clustered captions and videos for retrieval tasks [9]. Individual classifiers were trained to identify candidate objects, scenes and actions, and then a probabilistic graphical model was used to estimate the most likely semantic content (subject, verb, object) in the video [10]. Based on the content estimated and a template for description generation, a sentence was generated as the caption. Although this approach effectively breaks down and simplifies the problem, pre-defining a training set of relevant objects and actions for recognition and relying on a sentence template for content translation limits the model's flexibility to analyze variable-length videos with uncommon features and produce semantically diverse captions.

More recent work introduced the use of 2D-Convolutional Neural Networks (CNN) to extract features from frames in a video [11]. Mean-pooling can be applied to collapse these features and provide a single feature vector representation of the entire video clip, which will then be passed into an Long Short-Term Memory (LSTM) model for information decoding and caption generation [12]. However, due to the indiscriminate averaging of the feature vectors representing all frames, this approach does not consider the ordering of frames in a video and fails to utilize the rich temporal information underlying the evolving frames. To better exploit the temporal structure, Venugopalan et al. used a sequence-to-sequence model where a stacked LSTM takes as input the output of a CNN applied to each input frame and encodes the frames sequentially before a sentence description is generated word by word [13]. Also incorporating optical flow [14] between consecutive frames into the end-to-end LSTM model, this approach can handle a variable number of input video frames and exploits more of videos' temporal structure to learn a language model that automatically outputs the caption accordingly.

As the concept of "attention" started to gain popularity in deep learning, allowing neural networks to learn alignments between different modalities [15], attention-based Long Short-Term Memory (LSTM) models have been proposed for the task of video captioning to capture the most relevant temporal segments in the video sequence [3]. Xu et al. presented a Multimodal-Attention LSTM model where multiple encoding LSTMs model the temporal structure for different modalities (e.g. frame and motion) in the video and the decoding LSTM applies a soft attention mechanism to generate the sentence with attention from different modalities and their related elements [16]. Yao et al. used a 3D CNN to represent the short temporal dynamics in the video input along with a text-generation Recurrent Neural Network (RNN) that includes the attention mechanism to select the most relevant temporal information [4]. One limitation of traditional 3D CNNs is that 3D convolutions are expensive to compute and pretrained 3D-CNN models are limited relative to 2D CNNs.

Our work also builds on other recent developments in deep learning. One of them is P3D, which is used in state-of-the-art works for activity recognition [5]. P3D is a pseudo-3d residual network, where the pseudo comes from the fact that it decomposes 3D convolutions into a 2D spatial convolution and a 1D temporal convolution for more efficient computation. Further, this also enables use of pre-trained models from 2D-CNNs that are more ubiquitous than their 3D counterparts. Another is the Transformer [17], which has quickly become the model of choice in NLP. One work has applied Transformers to the task of dense-video captioning [18].

3 Approach

We explored several different approaches for video captioning. Our baseline model was a 2D-CNN with LSTM encoder and LSTM decoder. In addition, we explored 4 different models in this work: P3D encoder with LSTM decoder, an ensemble P3D/2D-CNN-LSTM encoder with P3D decoder, Transformer with 2D-CNN encoder, and Transformer with P3D encoder. To the best of our knowledge, we have not seen attention applied over P3D features with either an LSTM or Transformer nor have we seen a P3D/2D-CNN-LSTM ensemble encoder. In particular, our goal with using attention over P3D is to attend to features in both space and time, and not just focus on temporal attention for video captioning, as is usually done in previous works. A Transformer with 2D-CNN features has been

applied to dense video captioning [18], and we extend this idea to single caption videos for direct comparison.

3.1 2D-CNN-LSTM Encoder with LSTM decoder

The baseline method used a ResNet-152 that was pre-trained on ImageNet to extract features from frames in the video. For this model, we used a pre-trained model from Torchvision. We used the final layer of ResNet-152 before the classification layer as our feature representation, which has dimension of 2048. The two final hidden states from each layer were concatenated and a linear transformation was applied before feeding this in as the decoder LSTM's initial hidden state. The same was done for the cell states. We only fed this into the first layer of the decoder LSTM, while the second layer did not receive an initial hidden/cell state.

At every time step i, we computed an attention distribution using additive attention over the encoder hidden states based on the previous decoder hidden state:

$$e_i = v^T tanh(W_1 h_i + W_2 s) \in R \tag{1}$$

where $W_1 \in R^{d_3 \times d_1}$, $W_2 \in R^{d_3 \times d_2}$ are weight matrices and $v \in R^{d_3}$ is a weight vector; encoder hidden states $h_1, ..., h_N \in R^{d_1}$, decoder hidden state $s \in R^{d_2}$, and d_3 is a hyperparameter for the attention dimensionality.

The distribution was then used to compute a weighted combination of the encoder hidden states to get an attention feature vector. This feature vector was concatenated with the embedding for word i in the caption, and this concatenated vector served as input into our decoder LSTM. The two-layer decoder LSTM then produces a softmax distribution over the vocabulary, from which the next word can be predicted. We adapted code from Assignment 4 for the LSTM, but changed the attention mechanism (how and where it is computed) and expanded to two layers.

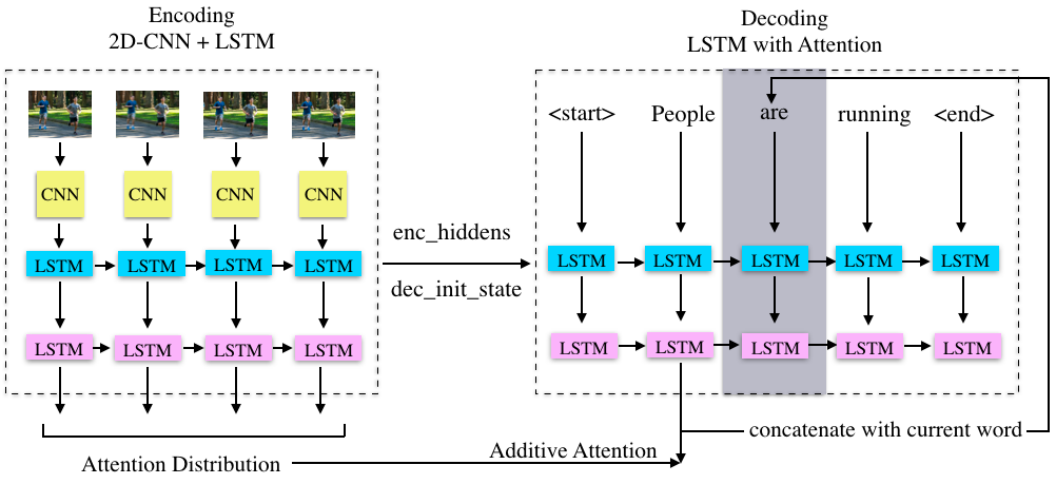

Figure 1: 2D-CNN-LSTM Encoder with LSTM Decoder: Training time architecture.

3.2 P3D Encoder with LSTM Decoder

In this model, we replace the 2D-CNN and LSTM Encoder module with P3D. The P3D model used was pre-trained on the Kinetics dataset for activity recognition (https://github.com/qijiezhao/pseudo-3d-pytorch), so there is strong reason to believe that this model can extract relevant features for the video captioning task. We used P3D to extract a 2x5x5x2048 (time, height, width, feature size) representation from the final convolutional layer before the last average pooling layer. The goal here was to capture attention over both time and space, since existing approaches focus primarily on temporal attention. Each 2048-vector is a feature representation, so P3D produced 50 feature vectors. Because P3D must take 16 frames as input, we applied P3D to non-overlapping sets of 16 consecutive frames and concatenate their feature representations.

A two-layer decoder LSTM then attends to these 2048-D feature representations through the same additive attention mechanism from the baseline model, where the weighted combination over these representations is then concatenated with word embedding as input into the LSTM at each step. For the initial hidden and cell states of the decoder LSTM, we averaged the P3D feature vectors and applied a linear transformation.

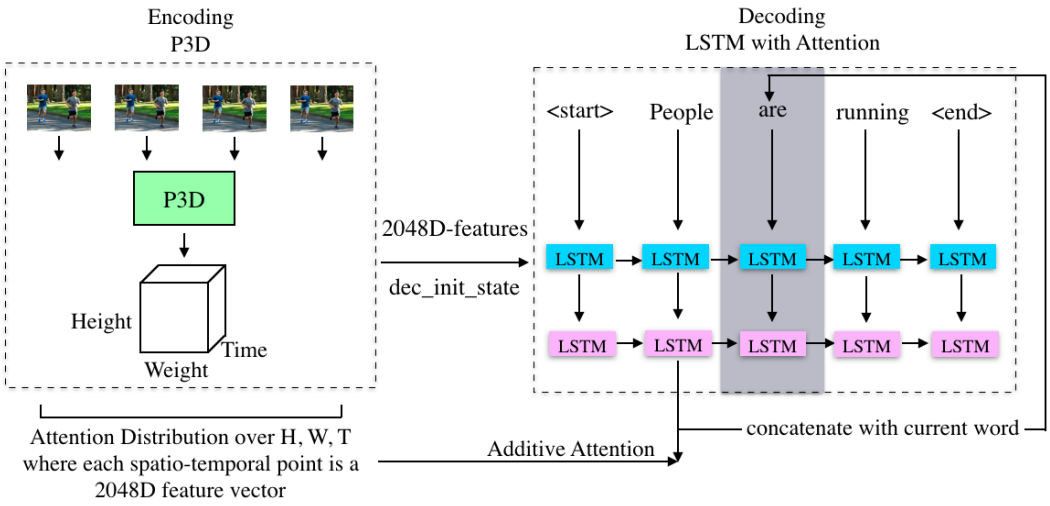

Figure 2: P3D Encoder with LSTM Decoder: Training time architecture.

3.3 Ensemble Encoder with LSTM Decoder

This approach combines the two above approaches for encoding into an ensemble encoding. We extract features with both a 2D-CNN-LSTM encoder and a P3D encoder, as described above. We derive the initial hidden and cell states for the decoder separately for each encoding method as done above, concatenate them, and then apply another linear transformation before inputting them into the decoder. We separately compute attention for each of the two encoded features (hidden states for the LSTM encoder and P3D features) based on the previous decoder hidden state. We then concatenate the two attention feature vectors with the word embedding and this three-way concatenation is the input into the decoder LSTM.

3.4 Transformer with 2D-CNN Features

In this fourth approach, we again use ResNet-152 to extract features for video frames. These features are then inputted into a Transformer encoder. A decoder Transformer uses the output from the encoder and the caption to predict the subsequent word for each word in the caption. We followed this guide (https://towardsdatascience.com/how-to-code-the-transformer-in-pytorch-24db27c8f9ec) for implementation of the Transformer, which closely follows the original Transformer paper. However, small modifications were made to better align with the latest practices. In particular, layer normalization is applied before each sub-layer rather than after the residual connection like was done in the original paper. A layer normalization is also applied on the final output of both encoder and decoder. We used 6 layers for both the encoder and decoder with a model dimension of 1024 and 16 heads for multi-head attention. The rest of implementation follows the original paper, including the same use of positional encoding, multi-head attention computation, and position-wise feed-forward layers [17].

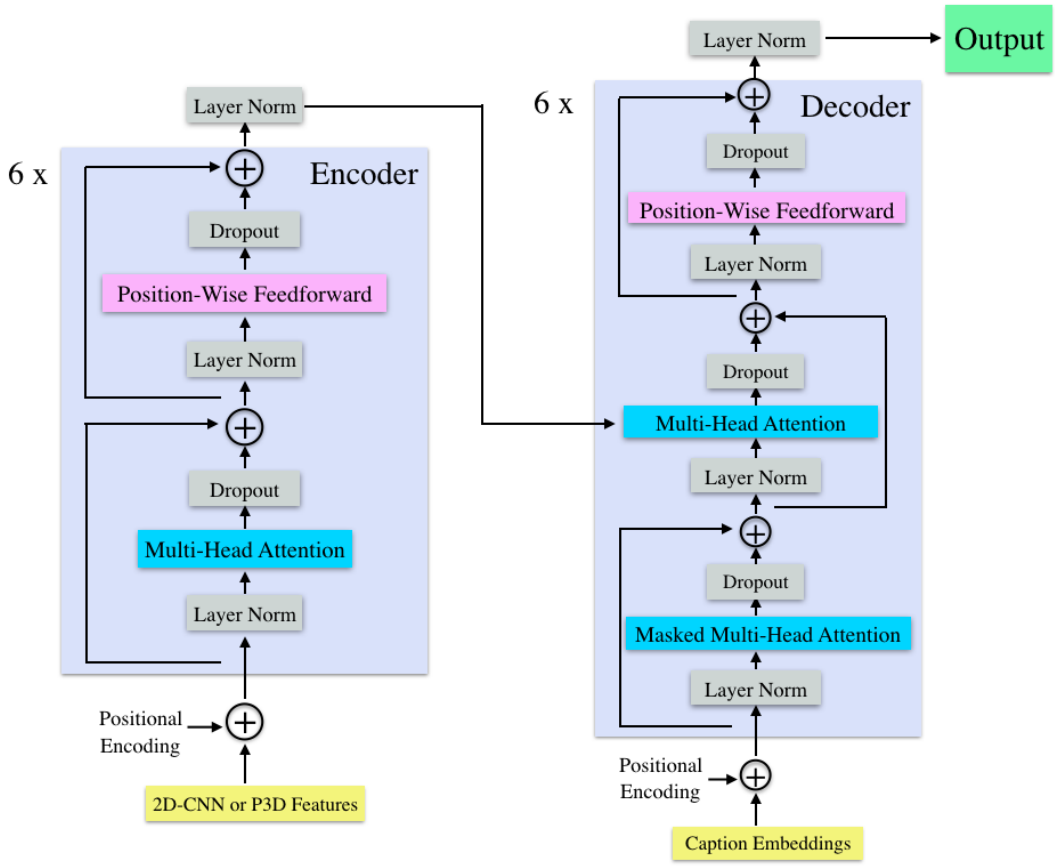

Figure 3: Transformer: Training time architecture.

3.5 Transformer with P3D Features

This final approach is similar to the previous approach, but instead of using a 2D-CNN to extract features from each frame, we used P3D to extract feature. We again extracted P3D features for non-overlapping sequences of 16 frames. These features were our input in the encoder module of the Transformer. The rest of the Transformer follows the implementation from the previous model.

4 Experiments

4.1 Data

We used the Microsoft Video Description Dataset (MSVD), which contains 1969 videos, each of which comes with multiple reference captions [19]. We followed the literature in using a split of 1200 for training, 100 for validation, and 669 for testing [19]. Each video comes with an average of 40 captions. For training, we choose 5 captions: the 5th longest caption to the 10th longest. We found that training on all of the captions made it difficult for the model to learn. Further, many captions were repetitive, and skewed the output towards the most common captions. The intuition behind using the 5th longest to the 10th longest is that shorter captions were often duplicated and longer captions were often very unique. Both of these could make it difficult for the model to generalize. To represent videos, we sampled every 10 frames, as done in previous work [13], with the exception that we sampled shorter sequences at a higher rate to ensure we had a minimum of 16 frames for P3D feature extraction.

4.2 Evaluation method

During training, we monitored the perplexity for the training and validation sets. For final evaluation, we used the BLEU metric for evaluation, which compares n-gram overlap between the hypothesized sentence produced from running beam search (beam size = 10) with our model to the corresponding reference sentences. We used the NLTK implementation of corpus bleu, as in the Assignment 4.

4.3 Experimental details

We experimented with hyperparameters on our baseline model. We used the Adam optimizer with the default beta parameters (0.9 and 0.999). Initially, we found that our model would output only a handful of different sentences, despite the loss/perplexity steadily decreasing, with a learning rate of 1e-4 and a batch size of 64/128. We ended up finding that a lower learning rate of 1e-5 and a small batch size (4) enabled optimal training. One potential reason for this is the skew in the dataset. Some words/phrases are far more represented than others ('man' vs 'woman' as an example), and many phrases/captions are more or less duplicated across videos. With a larger learning rate and bigger batch size, the model seemed to get stuck at a local minimum early on and struggled to get out. A smaller learning rate could help prevent the model from jumping to a local minimum too early on and a smaller batch could have contributed enough to noise to get out of non-ideal minimum. In particular, when using a big batch, the model may have trouble learning rarer words and phrases since the gradient gets diluted with more common words/phrases in the batch.

We converted all words to lower case and stripped punctuation. We also experimented with using GloVe Embeddings [20] to initialize our word vectors, using only the words that overlapped with the GloVe embeddings (8075 out of 9843 unique words in the training set). We found that Glove embeddings improved performance. We tried several approaches for incorporating the embeddings: fixing the GloVe embeddings (no fine-tuning), fine-tuning with a learning rate of 1e-5, and fine-tuning with a reduced learning rate of 1e-6. We found that fine-tuning with a reduced learning rate performed best. All other weights were initialized with Xavier Uniform initialization and all biases were initially set to 0.

We applied a weight decay of 1e-4 to prevent overfitting. For the LSTM models, we applied a dropout of 0.3 to the input video features and to the output of the LSTM decoder. For the Transformer models, we applied a dropout of 0.1 after each Layer Normalization. Finally, we optimized over the focal loss, which is a modification of the softmax loss that down-weights well-classified examples (probability of correct word is high) and focuses more on challenging examples [21]. To compute focal loss, the softmax loss was multiplied by a factor of $(1-p)^y$, where p is the probability of the correct class and y is a hyperparameter we set to 5.0. We trained for about 30 epochs for the LSTM models, and about 15 for the Transformer (due to time constraints).

$$FL(p_t) = -(1-p_t)^y log(p_t) \tag{2}$$

Due to time constraints, we used the same hyperparameters that we found worked best for our baseline model when training our other models. We note this as a limitation of our work.

4.4 Results

We show BLEU score results on both the train and the test set (Figures 4 and 5). We select the best model based on the lowest validation set perplexity, which we check every 1000 iterations. Overall, we found that P3D features did not perform as well as the ResNet-152 features (37.75 vs 33.51 for LSTM decoder and 40.78 vs 35.40 for Transformer). However, the ensemble model that used both ResNet-152 and P3D features with an LSTM decoder outperformed the ResNet-152 features alone (38.55 vs 37.75). While the results are not what we hoped, we believe that there are possible explanations that could be addressed in future work. For example, one potential issue with P3D is that the images used for pre-training (Kinetics dataset) may not well reflect the MSVD dataset we used. We did not fine-tune the P3D model because of the significant compute required, but doing so may help improve performance by making the features more relevant to this particular dataset. It's also possible that the sampling method used (every 10 frames) does not coincide well with how P3D is trained. Modifying the sampling procedure could thus improve performance. Finally, as described

in more detail in the analysis, P3D, being trained for activity recognition, may actually not be as well equipped to pick up on image features such as male vs. female.

We also found that Transformers performed better than LSTMs (40.78 vs 37.75 for ResNet-152 features and 35.40 vs. 33.51 for P3D features). One thing to note is that we did not have time to train the Transformer models as long as the LSTM models since they were our final experiments. We note that the train BLEU scores for the Transformer models are lower than their LSTM counterparts. It's possible that further training would help improve the BLEU scores on both the train and test sets since the model may not have yet fully completed learning.

BLEU Score	2D-CNN-LSTM Encoder + LSTM decoder	P3D Encoder + LSTM decoder	Ensemble Encoder + LSTM decoder	Transformer + 2D-CNN Features	Transformer + P3D Features
Train	71.85	64.11	66.73	61.47	54.25
Test	37.75	33.51	38.55	40.78	35.4

Figure 4: BLEU scores of models during train and test time.

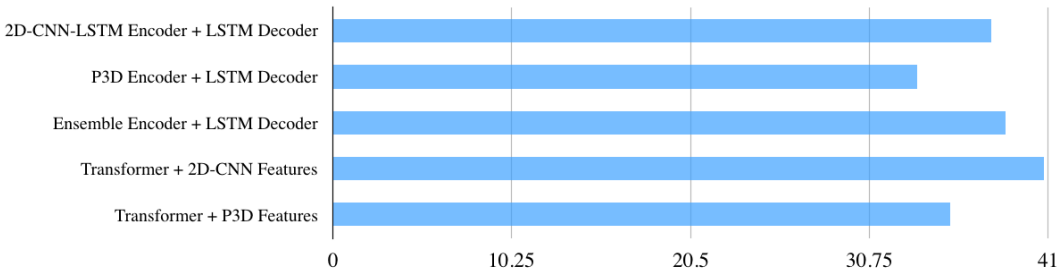

Figure 5: Comparison of BLEU scores of models at test time.

5 Analysis

One issue we ran into before finding the optimal hyperparameters on the baseline was that the model was learning to output only a small number of sentences, such as "the man is playing the guitar". While tuning the hyperparameters largely resolved this issue, we found that after looking through the dataset, there is a skew in the captions, with many captions being more or less repeated across videos. In particular, "the man is playing a guitar" is a common reference caption. Moreover, some words/phrases are much more represented than others (i.e."man/men" are more represented than "woman/women"). As a result, we decided to look at the BLEU score performance by subgroups.

For one comparison, we looked at performance on test captions that included "man" or "men" but not "woman" or "women" and vice versa. There were 291 captions that fell into the former and only 87 that fell into the latter. We also note that in the training set we had 541 for the "man/men" case and 152 in the "woman/women" case. We can see based on Figure 6 that for all models, the BLEU score performance is significantly higher in the "man/men" captions. Even for the best performing model, Transformer with 2D-CNN features, the gap is significant. Thus, even after tuning the optimization, there is still a skew in performance that reflects the skew in the dataset. The model often mistakes "woman" for "man", probably because it learns that "man" is simply more the more likely word than "woman". This also implies that the model is either not sufficiently conditioning on the extracted image features or that the features are not sufficiently capturing this information. One interesting observation is that performance on the "woman" subset is particularly bad for P3D, excluding the ensemble model. Because P3D is trained on activity recognition, it may not be well-equipped to discern things like gender since it is more tuned to picking up different activities. This could also be why the ensemble model shows an improvement over the baseline as the ResNet and P3D features could be capturing different information.

We also did a subgroup analysis on test cases that had the word "play" (or a close variation like "plays" or "playing") in at least one of its reference captions vs. cases that didn't. Out of 669 test cases, 209 had the word play in at least one reference. This was again motivated by the fact that we observed "play" frequently in our outputs. Overall, most models showed better performance on the cases where "play" was in one of the captions than in cases where "play" wasn't. One could argue that the model could get lucky by outputting a common word like "play" when it doesn't actually pick up on the larger context of the video. This is a reasonable explanation to explain the somewhat small, but still significant discrepancy in 3 out of the 5 models (2D-CNN-LSTM Encoder w/ LSTM decoder, Ensemble, and Transformer with P3D) as shown in Figure 7. This could mean that overall BLEU scores may be slightly inflated artificially by outputting common words without a true, specific understanding of the video. Further analysis into the performance with other common word subgroups would help corroborate this claim.

We show some example output, their references, and analysis in the appendix.

BLEU Score	2D-CNN-LSTM Encoder + LSTM decoder	P3D Encoder + LSTM decoder	Ensemble Encoder + LSTM decoder	Transformer + 2D-CNN Features	Transformer + P3D Features
"Man"	38.75	36.5	38.74	44.6	37.27
"Woman"	32.04	23.18	34.59	33.68	22.48

Figure 6: "Man" vs. "Woman".

BLEU Score	2D-CNN-LSTM Encoder + LSTM decoder	P3D Encoder + LSTM decoder	Ensemble Encoder + LSTM decoder	Transformer + 2D-CNN Features	Transformer + P3D Features
"Play"	39.33	32.64	40.26	40.91	37.46
No "Play"	36.67	33.95	37.3	40.43	34.21

Figure 7: "Play" vs. No "Play".

6 Conclusion

Overall, we did not find that P3D features performed better than 2D-CNN features derived from a ResNet-152 model. However, including P3D features along with ResNet features resulted in a modest improvement in performance. We also found that Transformers outperformed their attention-based LSTM counterparts, despite not having been able to train the Transformer models as long as the LSTM ones (due to it being a stretch goal we got in at the end).

We note the following limitations of our work and address areas for future improvement. For one, we were not able to tune hyperparameters for each model due to time constraints and applied the same hyperparameters to each model. Additionally, recent work has found that Faster-RCNN is often better for feature extraction [22], so one simple improvement would be to replace Faster-RCNN as our 2D-CNN feature extractor. Additionally, most state-of-the-art results incorporate additional information as input to their models, such as optical flow. Using an ensemble model with both optical flow and image features would likely improve results.

Although we did not achieve the results we had hoped for spatio-temporal attention with P3D, we were pleased to learn about and obtain results for diverse architectures. We believe that there is a lot of room for exploration that can build upon our work. Some potential changes, as mentioned in the results, would include fine-tuning (some) layers of P3D for this task and modifying the sampling rate to not skip so many frames. Another future approach is ensembling 2D-CNN and P3D features with a Transformer since we saw improvement in the LSTM case that could indicate that they may pick up on different, but relevant, features. Finally, it would be worthwhile to benchmark on other datasets. Many videos in this dataset are quite limited in motion so it is unsurprising the image features alone are sufficient for the task.

7 Additional Information

Mentor: Sahil Chopra

8 Appendix

Examples of Reference Captions	Caption by Ensemble Encoder + LSTM Decoder	Caption Quality
a man is playing on a piano someone is playing on a keyboard a musician playing on a piano	the man is playing the piano	High
two zebras are playing with each other two zebras are cuddling with each other a couple of zebras are rubbing heads	two zebras are walking in the grass	Medium
kids are walking in a parking lot four children crossing the road kids running to the store	a boy is running on a car	Low

Figure 8: Examples of video captions by Ensemble Encoder with LSTM decoder.

Analysis: We can see that the model is well able to learn certain sentences, such as "the man is playing the piano". This is a caption that is strongly represented in the training set and should be fairly easy to learn. The model can also learn slightly more challenging sentences that occur less frequently, such as picking up on two zebras. Although not exactly correct, the caption depicts a general, realistic understanding of the video. Finally, there are cases where the model is able to pick up on features like "boy" and "car" but cannot extract actual meaning or realistic activity in the video ("boy running on a car"). This implies that the model still has trouble establishing relationships between objects.

Examples of Reference Captions	Caption by Transformer with 2D-CNN Features	Caption Quality
one girl doing exercise a woman is exercising her abs a woman is doing exercise in the room	a woman is doing pushups	High
a man puts together a speaker a man is assembling a machine the man demonstrated the car speaker	a man is talking in the phone	Medium
a man turns on the microwave a man is cooking something the man heated a cup of coffee in the microwave	the man is cutting the something	Low

Figure 9: Examples of video captions by Transformer with 2D-CNN Features.

Analysis: For the first caption, the model is accurately able to pick up the subject ("woman") and the general activity of exercise. While "pushup" isn't in the reference, the model is still able to capture the general activity in the video. In the second caption, the model is able to discern the subject ("man") and its interaction with some electronic device. However, we can see that the model mistakes a "speaker" for a "phone", likely due to the fact that the word "phone" is more commonly seen as being close to a person than a "speaker". This issue could be probably be corrected by training on more diverse data. Finally, in the last caption, it seems that the model is only picking up on very vague context, such as being located in the kitchen, but cannot capture more detail than that, and thus outputs "cutting the something", a common kitchen activity.

References

[1] Song, J., Gao, L., Liu, L., Zhu, X., & Sebe, N. (2018). Quantization-based hashing: a general framework for scalable image and video retrieval. Pattern Recognition, 75, 175-187.

[2] Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2015). Show and tell: A neural image caption generator. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 3156-3164.

[3] Guo, Z., Gao, L., Song, J., Xu, X., Shao, J., & Shen, H.T. (2016). Attention-based LSTM with Semantic Consistency for Videos Captioning. ACM Multimedia.

[4] Yao, L., Torabi, A., Cho, K., Ballas, N., Pal, C.J., Larochelle, H., & Courville, A.C. (2015). Describing Videos by Exploiting Temporal Structure. 2015 IEEE International Conference on Computer Vision (ICCV), 4507-4515.

[5] Qiu, Z., Yao, T., & Mei, T. (2017). Learning Spatio-Temporal Representation with Pseudo-3D Residual Networks. 2017 IEEE International Conference on Computer Vision (ICCV), 5534-5542.

[6] https://merchdope.com/youtube-stats/

[7] Krizhevsky, A., Sutskever, I., & Hinton, G.E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Commun. ACM, 60, 84-90.

[8] Aradhye, H.B., Toderici, G., & Yagnik, J. (2009). Video2Text: Learning to Annotate Video Content. 2009 IEEE International Conference on Data Mining Workshops, 144-151.

[9] H. Huang, Y. Lu, F. Zhang, & S. Sun. A multi-modal clustering method for web videos. In ISCTCS. 2013. 2

[10] Thomason, J., Venugopalan, S., Guadarrama, S., Saenko, K., & Mooney, R.J. (2014). Integrating Language and Vision to Generate Natural Language Descriptions of Videos in the Wild. COLING.

[11] Krizhevsky, A., Sutskever, I., & Hinton, G.E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Commun. ACM, 60, 84-90.

[12] Venugopalan, S., Xu, H., Donahue, J., Rohrbach, M., Mooney, R.J., & Saenko, K. (2015). Translating Videos to Natural Language Using Deep Recurrent Neural Networks. HLT-NAACL.

[13] Venugopalan, S., Rohrbach, M., Donahue, J., Mooney, R.J., Darrell, T., & Saenko, K. (2015). Sequence to Sequence – Video to Text. 2015 IEEE International Conference on Computer Vision (ICCV), 4534-4542.

[14] Brox, T., Bruhn, A., Papenberg, N., & Weickert, J. (2004). High Accuracy Optical Flow Estimation Based on a Theory for Warping. ECCV.

[15] Luong, T., Pham, H.Q., & Manning, C.D. (2015). Effective Approaches to Attention-based Neural Machine Translation. EMNLP.

[16] Xu, J., Yao, T., Zhang, Y., & Mei, T. (2017). Learning Multimodal Attention LSTM Networks for Video Captioning. ACM Multimedia.

[17] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. NIPS.

[18] Zhou, L., Zhou, Y., Corso, J.J., Socher, R., & Xiong, C. (2018). End-to-End Dense Video Captioning with Masked Transformer. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 8739-8748.

[19] Xu, J., Mei, T., Yao, T., & Rui, Y. (2016). MSR-VTT: A Large Video Description Dataset for Bridging Video and Language. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 5288-5296.

[20] Pennington, J., Socher, R., & Manning, C.D. (2014). Glove: Global Vectors for Word Representation. EMNLP.

[21] Lin, T., Goyal, P., Girshick, R.B., He, K., & Dollár, P. (2017). Focal Loss for Dense Object Detection. 2017 IEEE International Conference on Computer Vision (ICCV), 2999-3007.

[22] Ren, S., He, K., Girshick, R.B., & Sun, J. (2015). Faster R-CNN: towards real-time object detection with region proposal networks.