Self-Supervised Learning with ConvLSTM AutoEncoder for Video Task Identification

Steve Han
stevehan2001@gmail.com

Anika Singh
anikasingh@utexas.edu

Grace Kim
yeeunk@utexas.edu

Abstract

As deep learning methods are becoming more and more capable, the amount of labeled data required to train a network is also increasing. Labeling data becomes the main hurdle of machine learning - especially for videos. With self-supervised learning, we can tap into the wealth of unlabeled videos that exist online to learn abstract features of those videos. We can then apply this knowledge to a more specific task. In this research, we are training a ConvLSTM AutoEncoder network to predict the next frame in a video. Afterward, we take the abstract knowledge learned during this process and apply it to an task classification. We will compare the results to a directly trained task classification network to evaluate the benefits of using self-supervised learning.

1 Introduction

Traditionally Convolutional Neural Networks (CNN) are used for video task recognition algorithms. However, CNNs require large amounts of labeled data to train on. Creating a method that doesn't require annotated data for videos is important. Not only can labeling data be expensive and time-consuming for researchers, but it can also be very susceptible to human errors. A video generally doesn't contain a singular action throughout the video, hence annotators must annotate each frame or set of frames which would cause a large variation in the annotations. [Fernando et al., 2017]. To address this problem, we use self-supervised learning that relies on significantly less labeled data as opposed to supervised learning. Self-supervised learning has two major components, a pretext task, and a downstream task.

Our methodology focuses on creating a robust model for task identification. Using self-supervised learning, we can reduce the amount of data being processed for the downstream task by having a pre-trained pretext task. The pretext task with unlabeled data allows us to thereby decrease our reliance on human-annotated data [Wang et al., 2019]. The pretext task allows for the model to learn meaningful features from the input which can be advantageous during training the downstream task.

Our project utilizes next frame prediction as a pretext task with a ConvLSTM AutoEncoder to learn features from the video. Our downstream task is the classification of our end task. Our dataset is composed of an air hockey game simulation with continuous movement throughout the game. This ensures that the next frame prediction algorithm learns meaningful features from the video. We also trained a classification task using our dataset to compare our proposed self-supervised model to a fully supervised model.

2 Backgrounds

2.1 Self-Supervised Learning

Self-supervised learning is a subset of unsupervised learning methods. It refers to learning methods in which ConvNets are explicitly trained with automatically generated labels. In his keynote speech at the AAAI conference, Yann LeCun described it as "learning to represent the world before learning a task. This is what babies and animals do... Once we have good representations of the world, learning

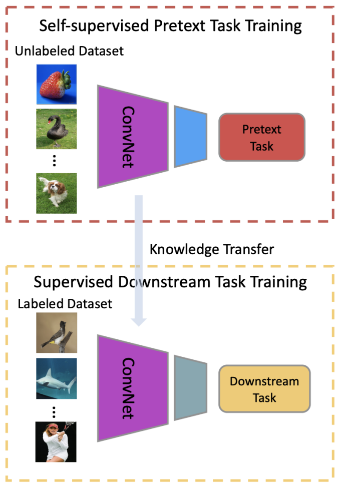


Figure 1: Self Supervised Learning Knowledge Transfer

a task requires few trials and few samples." The Figure 1 from [Jing and Tian, 2019] illustrates this process clearly.

For the pretext task, there are many popular options such as image colorization, image inpainting, and frame order prediction. The key is that all of these tasks are self-supervised, meaning that the label can be generated automatically instead of manually labeled by humans. We chose to use video future frame prediction since our downstream task is highly dependent on the ability to recognize where the objects are moving.

2.2 LSTM

For our network to process a video, it has to be able to look at multiple frames in a sequence. A basic model would be the Recurrent Neural Network (RNN), which is a network with loops in it, allowing information to persist. However, if we want to selectively remember information a long time in the past, a straightforward RNN isn't going to cut it. This is where LSTM comes in. LSTMs are explicitly designed to avoid the long-term dependency problem. This is an LSTM cell illustrated by Figure 2 [Olah, 2015].

The horizontal black line that runs through the top of the cell is the cell state. As information flows through the state, 3 gates in the cell make modifications to the state. The forget gate decides which information in the past to forget, the input gate

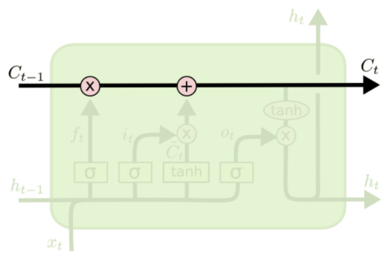


Figure 2: LSTM Cell

chooses the new data to add, and the output gate selects which states to output. Each gate is a sigmoid layer that outputs a number between 0 an 1 for each number in the cell state, which are then multiplied with the cell state.

2.3 Autoencoder

An autoencoder is a neural network structure that imposes a bottleneck that forces a compressed knowledge representation of the original input (Figure 3). By doing so, the neural network learns to pick up on the most important features of the input.

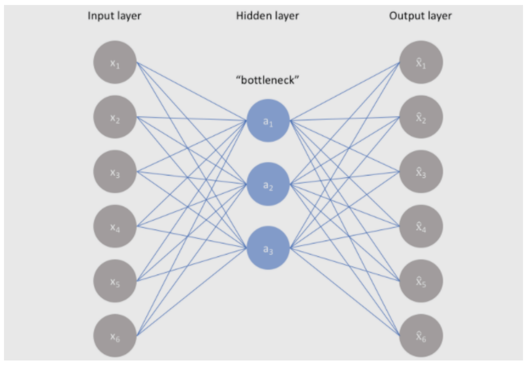
The ideal autoencoder model is sensitive to the inputs enough to accurately build a reconstruction, but not so much that the model simply memorizes or overfits the training data.

We believe that using an autoencoder is good for our use case since we want the self-supervised network to learn the most important features during the pretext task.

The decoder layer allows us to convert the abstract knowledge in the bottleneck layer into a tangible representation, which we can use to compute the loss. After the pretext training is done, we can detach the decoders and attach a classifier instead.

3 Related Works

Self Supervised Learning is increasingly becoming popular due to its reduced reliance on labeled data. A pretext task often alters the input data in some way and has the neural network determine the alteration. There exists current work which uses various different pretext tasks to increase the performance of self supervised learning models. The odd-ones-out network has been used as a pretext task for self-supervised learning [Fernando et al., 2017]. The



(Image by [Jordan, 2018])

Figure 3: Autoencoder Architecture

odd-ones-out network changes the order of video clips and the network determines the validity of the clips. This pretext task has been shown to produce 12.7 percent more accuracy on action classification tasks. Another example of a pretext task is the Pace Prediction where the alter the input data to be super slow, slow, normal, fast, and super fast [Wang et al., 2019]. Other examples of pretext tasks include rotation prediction of images or frames [Gidaris et al., 2018].

Another paper studying self supervised learning uses motion and appearance statistics predictions has shown how self supervised learning is advantageous [Wang et al., 2020]. This paper takes statistical concepts such as fast-motion and dominant color and pretrains the model using these features.

4 Approach

4.1 Pretext task

The goal of the pretext task is to predict the next frame of the video with information from previous frames using ConvLSTM Autoencoder [Shi et al., 2015], a variant of LSTM (Long Short-Term Memory). As we stated, we use LSTM, which is a type of RNN. Traditionally, RNN has proven to have great success with tasks such as video frame prediction because of its ability to deal with sequential data such as video representation. Because our dataset usually consists of a large number of frames, we decided to adapt LSTM over simple RNN to overcome RNN's limitation with long-term representation such as vanishing and exploding gradient. [Oprea et al., 2020] Also, it's necessary to keep the information from the previous frames for our case as the model's task is essentially to predict the motion happening in the scene of the video.

4.2 Downstream task

The downstream task is to classify the video's task, which is the destination of the puck: left and right, given the single frame of the video. Using a self-supervised learning approach, we share the encoder from the pretext task with its information learned from training. After the encoder, we add one hidden layer and a softmax classifier to output the label. Choosing softmax over sigmoid wouldn't make a big difference since we only have left and right as the output, which is a binary classification. However, we still use the softmax classifier for later adaption to multi-class classification tasks by adding more labels, the puck's direction.

4.3 Model

We adapt the model from [Nielsen, 2020]. The model utilizes a Seq2Seq model [Sutskever et al., 2014; Cho et al., 2014] which is a model trained by converting sequences from one domain to another domain. Although the Seq2Seq model is more commonly used in NLP tasks, we can still benefit from it by simply replacing the layers in the encoder and decoder with the ConvLSTM layers, which replace matrix multiplication with convolution operation at each gate in the LSTM cell. This way, it can deal with images instead of texts.

Respectively, encoder and decoder contain two ConvLSTM layers. For the pretext task, we attach a decoder and 3D CNN layer. The decoder in the pretext task will output feature map reconstructed from the encoded representation, and the 3D CNN layer takes that output to construct the image using a 3D kernel. After rendering the output with 3D CNN layer, we compute the loss based on the difference between the pixel value of the prediction and the actual frame. Downstream task has one hidden layer and softmax classifier, attached after the shared encoder, to output the destination of the hockey puck.

4.4 Training

Training our model has two essential parts: pretext and downstream. To train the pretext task, we predict the next frame, as described in section

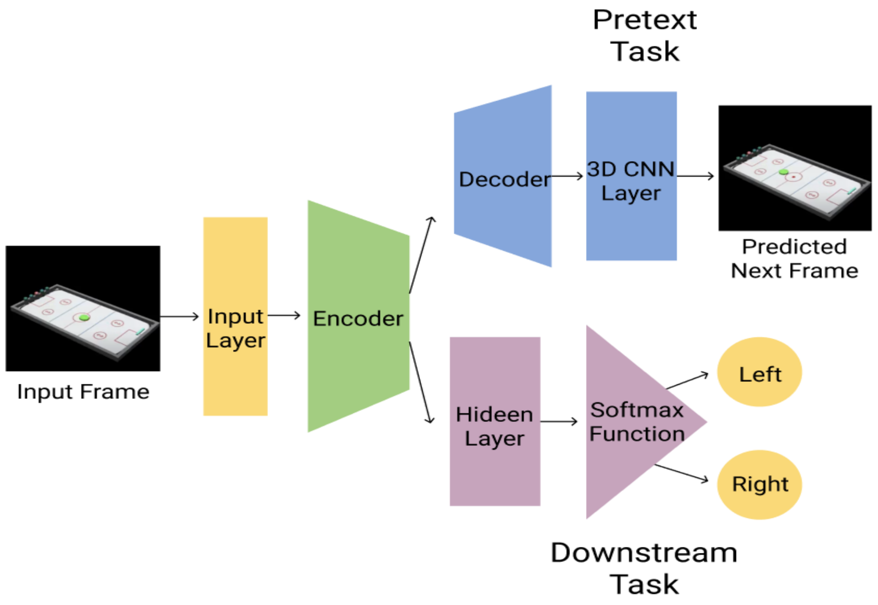


Figure 4: Model Architecture

4.3. The pretext task training uses a Mean Squared Error (MSE) loss with the input frame and the output frame's pixel values. The downstream task uses a Cross-Entropy as its loss function. The advantage of self-supervised learning is that the downstream classification task doesn't require as much data to train as classifying without a pretext task. The downstream task training uses the features learned from the pretrained pretext task. We transfer this knowledge by sharing the pretext task's encoder with downstream task training. We freeze the downstream model, allowing the new task classification head to be calibrated with the rest of the model. Freezing the model means the weights will not be adjusted while backward passing and thereby decreasing training time. Afterward, we unfreeze the model to allow the downstream task to continue training. Freezing layers of the model technique is inspired by the fact that in some cases the earlier layers, which have fewer parameters, take up more time as compared to later layers, with more parameters. [Brock et al., 2017]

5 Data

Our dataset is the recorded videos of an air hockey puck moving on an air hockey table (courtesy of Russel Coleman for the environment). The environment is powered by MuJoCo, a robotics simulation framework commonly used for reinforcement learning. The puck starts at the end of the table that is opposite to the five red buttons. When hit with a puck, each button turns from red to green. Our dataset only contains the video of the puck hitting either the leftmost or rightmost button for the sake of simplicity.

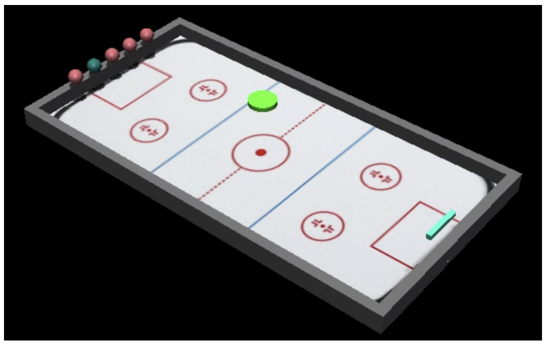


Figure 5: Air Hockey Table Simulation

Together with the environment, we have a Reinforcement learning model trained with the goal of hitting the puck to the buttons. We can use this model to generate datasets automatically. To add variation to the data, the puck starts at a random position, and the reinforcement learning model moves the neon-green bar to hit the puck toward a button. If the puck hits the left or right button, the video recording is saved and labeled automatically as either left or right.

6 Evaluation

For the purpose of comparison, we also implement a supervised learning approach where the network is asked to perform the same downstream task as the self-supervised model. We will use the same model, but instead of borrowing the knowledge from the pretext task, it is directly trained on the downstream task. We compare the results of these 2 approaches in terms of how accurate the classifications are and how many frames each needs to predict the task. We are also interested to see if the self-supervised approach speeds up the convergence rate of training.

Since we weren't able to get the dataset on time, we don't have any results at the moment. However, we will have the research completed in the next two weeks, so stay tuned for our final presentation!

References

[Brock et al., 2017] Brock, A., Lim, T., Ritchie, J. M., and Weston, N. (2017). Freezeout: Accelerate training by progressively freezing layers.

[Cho et al., 2014] Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation.

[Fernando et al., 2017] Fernando, B., Bilen, H., Gavves, E., and Gould, S. (2017). Self-supervised video representation learning with odd-one-out networks.

[Gidaris et al., 2018] Gidaris, S., Singh, P., and Komodakis, N. (2018). Unsupervised representation learning by predicting image rotations.

[Jing and Tian, 2019] Jing, L. and Tian, Y. (2019). Self-supervised visual feature learning with deep neural networks: A survey.

[Jordan, 2018] Jordan, J. (2018). Introduction to autoencoders. `https://www.jeremyjordan.me/autoencoders/`.

[Nielsen, 2020] Nielsen, A. H. (2020). Video prediction using convlstm autoencoder (pytorch). `https://github.com/holmdk/Video-Prediction-using-PyTorch`.

[Olah, 2015] Olah, C. (2015). Understanding lstm networks. `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

[Sutskever et al., 2014] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks.

[Wang et al., 2019] Wang, J., Jiao, J., Bao, L., He, S., Liu, Y., and Liu, W. (2019). Self-supervised spatio-temporal representation learning for videos by predicting motion and appearance statistics.

[Wang et al., 2020] Wang, J., Jiao, J., and Liu, Y.-H. (2020). Self-supervised video representation learning by pace prediction.