Unsupervised Learning of Video Representations using LSTMs

Nitish Srivastava NITISH@CS.TORONTO.EDU
Elman Mansimov EMANSIM@CS.TORONTO.EDU
Ruslan Salakhutdinov RSALAKHU@CS.TORONTO.EDU
University of Toronto, 6 Kings College Road, Toronto, ON M5S 3G4 CANADA

Abstract

We use Long Short Term Memory (LSTM) networks to learn representations of video sequences. Our model uses an encoder LSTM to map an input sequence into a fixed length representation. This representation is decoded using single or multiple decoder LSTMs to perform different tasks, such as reconstructing the input sequence, or predicting the future sequence. We experiment with two kinds of input sequences – patches of image pixels and high-level representations ("percepts") of video frames extracted using a pretrained convolutional net. We explore different design choices such as whether the decoder LSTMs should condition on the generated output. We analyze the outputs of the model qualitatively to see how well the model can extrapolate the learned video representation into the future and into the past. We further evaluate the representations by finetuning them for a supervised learning problem – human action recognition on the UCF-101 and HMDB-51 datasets. We show that the representations help improve classification accuracy, especially when there are only few training examples. Even models pretrained on unrelated datasets (300 hours of YouTube videos) can help action recognition performance.

1. Introduction

Understanding temporal sequences is important for solving many problems in the AI-set. Recently, recurrent neural networks using the Long Short Term Memory (LSTM) architecture have been used successfully to perform various supervised sequence learning tasks, such as speech recognition (Graves & Jaitly, 2014), machine translation (Sutskever et al., 2014; Cho et al., 2014), and caption generation for images (Vinyals et al., 2014). They have also been applied on videos for recognizing actions and generating natural language descriptions (Donahue et al., 2014). A general sequence to sequence learning framework was described by Sutskever et al. (2014) in which a recurrent network is used to encode a sequence into a fixed length representation, and then another recurrent network is used to decode a sequence out of that representation. In this work, we apply and extend this framework to learn representations of sequences of images. We choose to work in the *unsupervised* setting where we only have access to a dataset of unlabelled videos.

1.1. Why Unsupervised Learning?

Supervised learning has been extremely successful in learning good visual representations that not only produce good results at the task they are trained for, but also transfer well to other tasks and datasets. Therefore, it is natural to extend the same approach to learning video representations. This has led to research in 3D convolutional nets (Ji et al., 2013; Tran et al., 2014), different temporal fusion strategies (Karpathy et al., 2014) and exploring different ways of presenting visual information to convolutional nets (Simonyan & Zisserman, 2014a). However, videos are much higher dimensional entities compared to single images. Therefore, it becomes increasingly difficult to do credit assignment and learn long range structure, unless we collect much more labelled data or do a lot of feature engineering (for example computing the right kinds of flow features) to keep the dimensionality low. The costly work of collecting more labelled data and the tedious work of doing more clever engineering can go a long way in solving particular problems, but this is ultimately unsatisfying as a machine learning solution. This highlights the need for using unsupervised learning to find and represent structure in videos. Moreover, videos have a lot of structure in them (spatial and temporal regularities) which makes them particularly well suited as a domain for building unsupervised learning models.

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

1.2. Our Approach

In this paper, we use the LSTM Encoder-Decoder framework to learn video representations. The Encoder LSTM runs through a sequence of frames to come up with a representation. This representation is then decoded through another LSTM to produce a target sequence. We consider different choices of the target sequence. One choice is to predict the same sequence as the input. The motivation is similar to that of autoencoders – we wish to capture all that is needed to reproduce the input but at the same time go through the inductive biases imposed by the model. Another option is to predict the future frames. Here the motivation is to learn a representation that extracts all that is needed to extrapolate the motion and appearance beyond what has been seen. These two natural choices can also be combined. In this case, there are two decoder LSTMs – one that decodes the representation into the input sequence and another that decodes the same representation to predict the future.

The inputs to the model can, in principle, be any representation of individual video frames. However, for the purposes of evaluation, we limit our attention to two kinds of inputs. The first is image patches. For this we use natural image patches as well as a dataset of moving MNIST digits. The second is the high-level "percepts" extracted by applying a convolutional net pretrained on ImageNet. These percepts are the states of the last (and/or second-to-last) layer of rectified linear hidden units.

In order to evaluate the learned representations we qualitatively analyze the reconstructions and predictions made by the model. For a more quantitative evaluation, we use these LSTMs as initializations for the supervised task of action recognition. If the unsupervised learning model comes up with useful representations then the classifier should be able to perform better, especially when there are only a few labelled examples. We find that this is indeed the case.

1.3. Related Work

The first approaches to learning representations of videos in an unsupervised way were based on ICA (van Hateren & Ruderman, 1998; Hurri & Hyvärinen, 2003). Mobahi et al. (2009) proposed a regularizer that encourages temporal coherence using a contrastive hinge loss. Le et al. (2011) approached this problem using multiple layers of Independent Subspace Analysis modules. Generative models for understanding transformations between pairs of consecutive images are also well studied (Memisevic, 2013; Memisevic & Hinton, 2010; Susskind et al., 2011). This work was extended recently by Michalski et al. (2014) to model longer sequences.

Recently, Ranzato et al. (2014) proposed a generative model for videos. The model uses a recurrent neural network to predict the next frame or interpolate between frames. This work highlights the importance of choosing the right loss function. It is argued that squared loss in input space is not the right objective because it does not respond well to small distortions in input space. The proposed solution is to quantize the image patches into a large dictionary and train the model to predict the identity of the target patch. This does solve some of the problems of squared loss but it introduces an arbitrary dictionary size into the picture and altogether removes the idea of patches being similar or dissimilar to one other. Other metrics such as Structural Similarity (Wang et al., 2004) have also been proposed. Designing a loss function that respects our notion of visual similarity is a very hard problem (in a sense, almost as hard as the modeling problem we want to solve in the first place). Therefore, in this paper, we use the simple squared loss objective function as a starting point and focus on designing an encoder-decoder RNN architecture that can be used with any differentiable loss function.

2. Model Description

In this section, we describe several variants of our LSTM Encoder-Decoder model. The basic unit of our network is the LSTM cell block that consists of four input terminals, a memory cell and an output unit. Our implementation of LSTMs follows closely the one discussed by Graves (2013).

2.1. LSTM Autoencoder Model

This model consists of two Recurrent Neural Nets, the encoder LSTM and the decoder LSTM as shown in Fig. 1. The input to the model is a sequence of vectors (image patches or features). The encoder LSTM reads in this sequence. After the last input has been read, the cell state and output state of the encoder are copied over to the decoder LSTM. The decoder outputs a prediction for the target sequence. The target sequence is same as the input sequence, but in reverse order. Reversing the target sequence makes the optimization easier because the model can get off the ground by looking at low range correlations. This is also inspired by a stack representation of lists (for example in LISP). The encoder creates a list by pushing frames on top of the stack and the decoder unrolls this list by removing frames from the top.

The decoder can be conditional or unconditioned. A conditional decoder receives the last generated output frame as input, i.e., the dotted boxes in Fig. 1 are present. An unconditioned decoder does not receive that input. This is discussed in more detail in Sec. 2.3. The architecture can be extended to multiple layers by stacking LSTMs on top of each other.

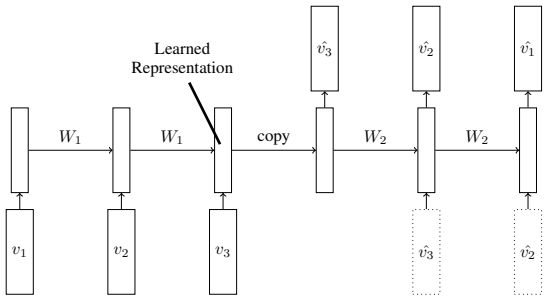


Figure 1. LSTM Autoencoder Model

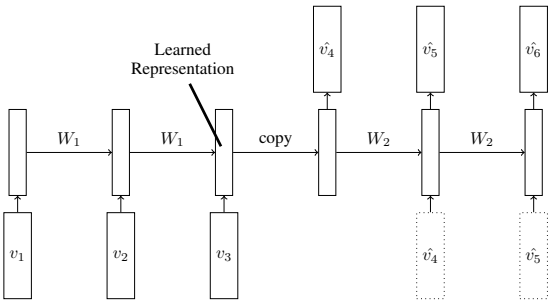


Figure 2. LSTM Future Predictor Model

Why should this learn good features ?
The state of the encoder LSTM after the last input frame has been read is the representation of the input video. The decoder LSTM is being asked to reconstruct back the input sequence from this representation. In order to do so, the representation must retain information about the appearance of the objects and the background as well as any motion contained in the video. This is exactly the information that we would like the representation to contain. However, an important question for any autoencoder-style model is what prevents it from learning a trivial identity mapping by effectively copying the input to the output. Two factors prevent this behaviour. First, the fact that there are only a fixed number of hidden units makes it unlikely that the model can learn trivial mappings for arbitrary length input sequences. Second, the same dynamics must be applied recursively on the representation. This further makes it hard for the model to learn an identity mapping.

2.2. LSTM Future Predictor Model

Another natural unsupervised learning task for sequences is predicting the future. This is the approach used in language models. The design of the Future Predictor Model is same as that of the Autoencoder Model, except that the decoder LSTM in this case predicts frames of the video that come just after the input sequence (Fig. 2). Ranzato et al. (2014) use a similar model but predict only the next frame at each time step. This model, on the other hand, predicts a long sequence into the future. Here again we consider two variants of the decoder – conditional and unconditioned.

Why should this learn good features ?
In order to predict the next few frames correctly, the model needs information about which objects are present and how they are moving so that the motion can be extrapolated. The hidden state coming out from the encoder will try to capture this information.

2.3. Conditional Decoder

For each of these two models, we consider two possibilities - one in which the decoder LSTM is conditioned on the last generated frame and the other in which it is not. In the experimental section, we explore these choices quantitatively. Here we briefly discuss the modeling issues involved. A conditional decoder helps model multiple modes in the target sequence distribution. Without that, we would end up averaging the multiple modes in the low-level input space. In order to fully exploit such a conditional decoder, we would need some stochasticity in the way we generate a frame. In the case of machine translation (Sutskever et al., 2014) this was achieved by sampling from the predicted multinomial distribution over words. Analogously, we could consider adding noise to the predicted frames in our model. However, simple ways of doing this, for example, adding independent Gaussian noise are unlikely to be effective since they only serve to corrupt the data and take it away from the data manifold. We need to stay on the manifold in order to generate plausible alternatives. For example, if based on some input video, a ball is equally likely to move left or right, we need a stochastic process that can predict the ball moving either to the left or to the right but not a ball in the middle plus Gaussian noise. This requires adding some noise in the higher layers of the model and then having a deterministic process generate the frame. Variational autoencoders and related methods (Kingma & Welling, 2013; Gregor et al., 2015) are promising ways of doing this. We intend to explore these techniques in future. In this work, however, we use a completely deterministic conditional decoder. At test time we feed in the generated frame from the previous step without adding any noise. At training time we feed in the ground truth.

Modeling multiple modes is an issue only if we expect multiple modes in the target sequence distribution. For the LSTM Autoencoder, there is only one correct target and hence the target distribution can be considered almost unimodal. But for the LSTM Future Predictor there is a possibility of multiple targets given an input. It should be noted that for videos the source of uncertainty about the future is often completely external to the input. For example, there is often no way to predict that a new object might come into the scene, or what kind of background will come into view as the camera moves. Therefore the future is reasonably predictable only for a very short time, making multiple modes in the target distribution less of a concern.

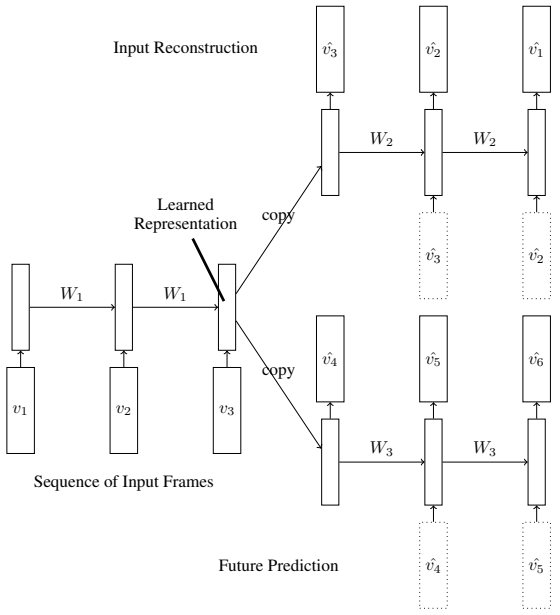


Figure 3. The Composite Model: The LSTM predicts the future as well as the input sequence.

2.4. A Composite Model

The two tasks – reconstructing the input and predicting the future can be combined to create a composite model as shown in Fig. 3. Here the encoder LSTM is asked to come up with a state from which we can *both* predict the next few frames as well as reconstruct the input.

This composite model tries to overcome the shortcomings that each model suffers on its own. A high-capacity autoencoder would suffer from the tendency to learn trivial representations that just memorize the inputs. However, this memorization is not useful at all for predicting the future. Therefore, the composite model cannot just memorize information. On the other hand, the future predictor suffers form the tendency to store information only about the last few frames since those are most important for predicting the future, i.e., in order to predict v_t, the frames $\{v_{t-1}, \ldots, v_{t-k}\}$ are much more important than v_0, for some small value of k. Therefore the representation at the end of the encoder will have forgotten about a large part of the input. But if we ask the model to also predict *all* of the input sequence, then it cannot just pay attention to the last few frames.

3. Experiments

We design experiments to accomplish the following objectives:

- Get a qualitative understanding of what the LSTM learns to do.
- Measure the benefit of initializing networks for supervised learning tasks with the weights found by unsupervised learning, especially with very few training examples.
- Compare the different proposed models – Autoencoder, Future Predictor and Composite models and their conditional variants.
- Compare with state-of-the-art action recognition benchmarks.

3.1. Training

The proposed models were trained by backpropagation. RMSProp gave much faster convergence than well-tuned stochastic gradient descent with momentum. More details about the training, weight initialization and other hyperparameters can be found in the expanded version of this paper (Srivastava et al., 2015).

3.2. Datasets

We use the UCF-101 and HMDB-51 datasets for supervised tasks. The UCF-101 dataset (Soomro et al., 2012) contains 13,320 videos with an average length of 6.2 seconds belonging to 101 different action categories. The dataset has 3 standard train/test splits with the training set containing around 9,500 videos in each split (the rest are test). The HMDB-51 dataset (Kuehne et al., 2011) contains 5100 videos belonging to 51 different action categories. Mean length of the videos is 3.2 s. This also has 3 train/test splits with 3570 videos in the training set and rest in test.

To train the unsupervised models, we used a subset of the YouTube videos from the Sports-1M dataset (Karpathy et al., 2014). Even though this dataset is labelled for actions, we did not do any supervised experiments on it because of logistical constraints with working with such a huge dataset. We instead collected 300 hours of video by randomly sampling 10 second clips. We also used the supervised datasets (UCF-101 and HMDB-51) for unsupervised training. However, we found that using them did not give any significant advantage over just using the YouTube videos. Percepts were extracted using the convolutional neural net model of Simonyan & Zisserman (2014b). The videos have a resolution of 240 $\times$ 320 and were sampled at 30 frames per second. The central 224 $\times$ 224 patch from each frame was forward proped to obtained the RGB percepts. We used only a single patch for simplicity of doing the experiments, although the performance can probably be improved by taking multiple patches, doing horizontal flips and other distortions. We also computed flow percepts by training the temporal stream convolutional network as described by Simonyan & Zisserman (2014a). We found that the fc6 features worked better than fc7 for single frame classification using both RGB and flow percepts. Therefore, we used the 4096-dimensional fc6 layer as the input representation of our data. Besides these percepts, we also

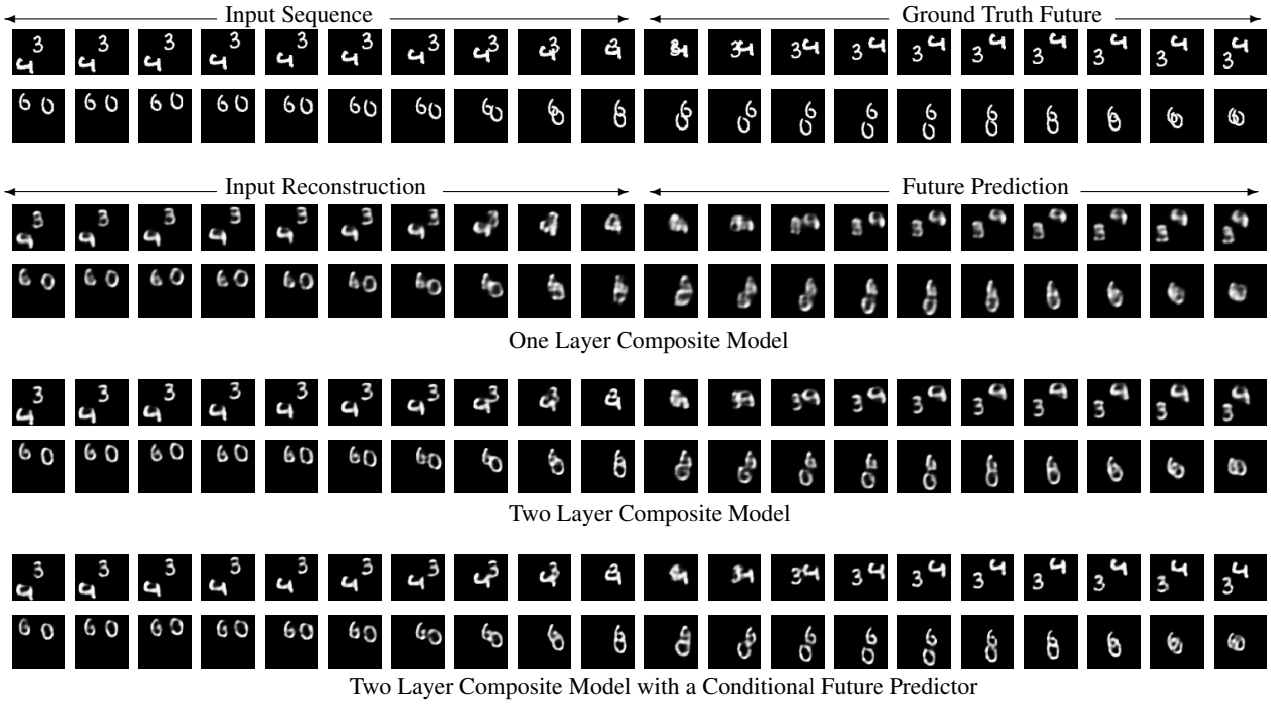


Figure 4. Reconstruction and future prediction obtained from the Composite Model on a dataset of moving MNIST digits.

trained the proposed models on 32 $\times$ 32 patches of pixels.

3.3. Visualization and Qualitative Analysis

The aim of this set of experiments is to visualize the properties of the proposed models. We first trained the models on a dataset of moving MNIST digits. Each video was 20 frames long and consisted of 2 digits moving inside a 64 $\times$ 64 patch. The digits were chosen randomly from the training set and placed initially at random locations inside the patch. Each digit was assigned a velocity whose direction was chosen uniformly randomly on a unit circle and whose magnitude was also chosen uniformly at random over a fixed range. The digits bounced-off the edges of the 64 $\times$ 64 frame and overlapped if they were at the same location. The reason for working with this dataset is that it is infinite in size and can be generated quickly on the fly. This makes it possible to explore the model without expensive disk accesses or overfitting issues. Even though it is simple to generate, this dataset has non-trivial properties because the digits occlude each other and bounce off walls.

We first trained a one layer Composite Model. The LSTM had 2048 units. The encoder took 10 frames as input. The decoder tried to reconstruct these 10 frames and the future predictor attempted to predict the next 10 frames. We used logistic output units with a cross entropy loss function. Fig. 4 shows two examples of running this model. The true sequences are shown in the first two rows. The next two rows show the reconstruction and future prediction from the one layer Composite Model. The model figures out how to separate superimposed digits and can model them as they pass through each other. This shows some evidence of *disentangling* the two independent factors of variation in this sequence. The model can also correctly predict the motion after the digits bounce off the walls. In order to see if adding depth helps, we trained a two layer Composite Model, with each layer having 2048 units. We can see that adding depth helps the model make better predictions. Next, we changed the future predictor by making it conditional. We can see that this model makes even better predictions. More experiments and analysis, including visualization of learned features and evolution of the LSTMs state can be found in the expanded version of this paper (Srivastava et al., 2015).

Next, we tried to see if our models can also work with natural image patches. For this, we trained the models using a conditional future predictor on sequences of 32 $\times$ 32 image patches extracted from the UCF-101 dataset. In this case, we used linear output units and the squared error loss function. The input was 16 frames. The model was asked to reconstruct these 16 frames and predict the future 13 frames. Fig. 5 shows the reconstructions obtained from a two layer Composite model with 2048 units. We found that the future predictions quickly blur out but the input reconstructions look better. We then trained a bigger model with 4096 units. Even in this case, the future blurred out quickly. However, the reconstructions look sharper. We believe that models that look at bigger contexts and use more powerful stochastic decoders are required to get better future predictions.

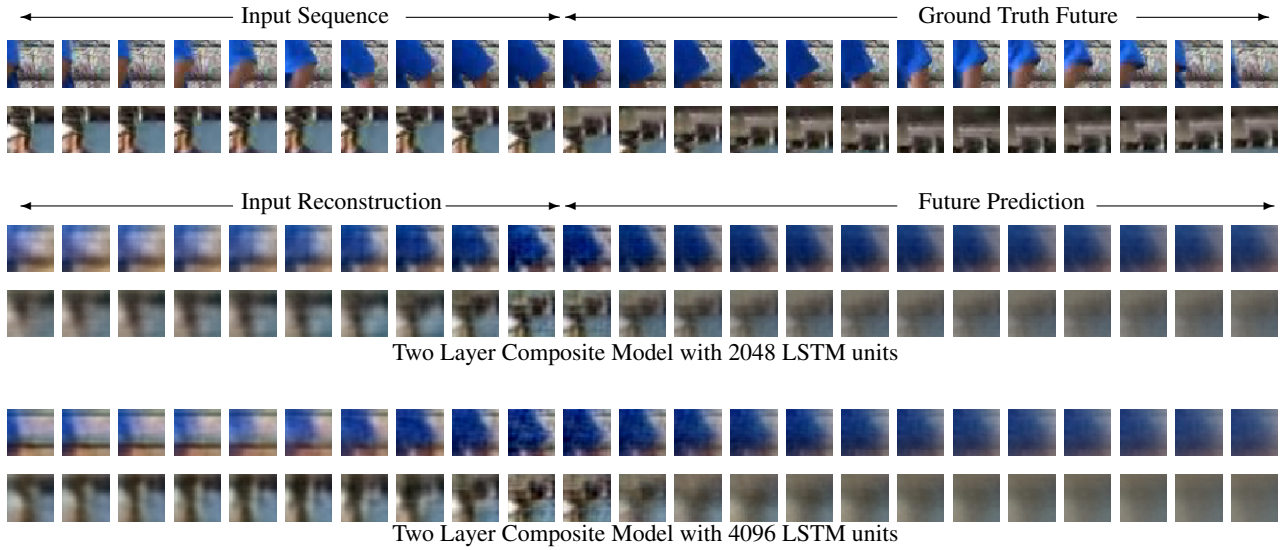


Figure 5. Reconstruction and future prediction obtained from the Composite Model on a dataset of natural image patches. The first two rows show ground truth sequences. The model takes 16 frames as inputs. Only the last 10 frames of the input sequence are shown here. The next 13 frames are the ground truth future. In the rows that follow, we show the reconstructed and predicted frames for two instances of the model.

3.4. Action Recognition on UCF-101/HMDB-51

The aim of this set of experiments is to see if the features learned by unsupervised learning can help improve performance on supervised tasks.

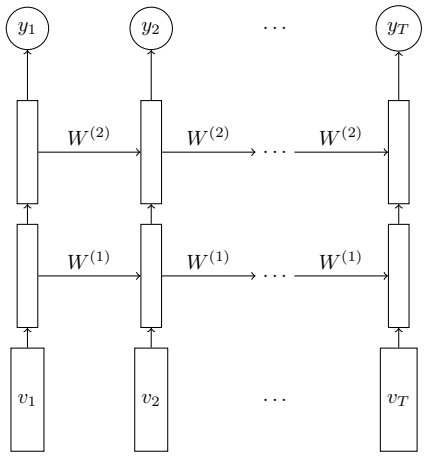
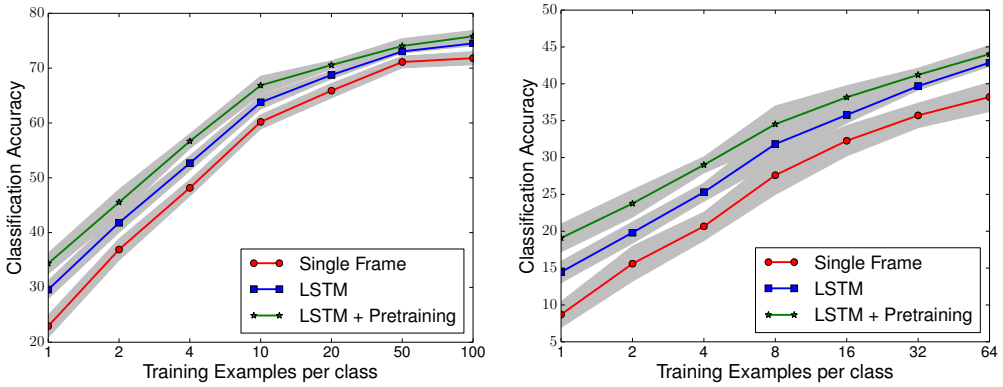


Figure 6. LSTM Classifier.

We used a two layer Composite Model with 2048 hidden units with no conditioning on either decoders. The model was trained on percepts extracted from 300 hours of YouTube data. It was trained to autoencode 16 frames and predict the next 13 frames. We initialize an LSTM classifier with the weights learned by the encoder LSTM from this model. The model is shown in Fig. 6. The output from each LSTM goes into a softmax classifier that makes a prediction about the action being performed at each time step. Since only one action is being performed in each video in the datasets we consider, the target is the same at each time step. At test time, the predictions made at each time step are averaged. To get a prediction for the entire video, we average the predictions from all 16 frame blocks in the video with a stride of 8 frames. Using a smaller stride did not improve results.

The baseline for comparing these models is an identical LSTM classifier but with randomly initialized weights. All classifiers used dropout regularization, where we dropped activations as they were communicated across layers but not through time within the same LSTM as proposed in Zaremba et al. (2014). We emphasize that this is a very strong baseline and does significantly better than just using single frames. Using dropout was crucial in order to train good baseline models with very few training examples.

Fig. 7 compares three models - single frame classifier, baseline LSTM classifier and the LSTM classifier initialized with weights from the Composite Model. The number of labelled videos per class is varied. Note that having one labelled video means having many labelled 16 frame sequences. We can see that for the case of very few training examples, unsupervised learning gives a substantial improvement. For example, for UCF-101, the performance improves from 29.6% to 34.3% when training on only one labelled video. As the size of the labelled dataset grows, the improvement becomes smaller. Even for the full UCF-101 dataset we get a considerable improvement from 74.5% to 75.8%. On HMDB-51, the improvement is from 42.8% to 44.0% for the full dataset (70 videos per class) and 14.4% to 19.1% for one video per class. Although, the improvement in classification by using unsupervised learning was not as big as we expected, we still managed to yield an additional improvement over a strong baseline.



(a) UCF-101 RGB (b) HMDB-51 RGB

Figure 7. Effect of pretraining on action recognition with change in the size of the labelled training set. The error bars are over 10 different samples of training sets.

Model	UCF-101 RGB	UCF-101 1- frame flow	HMDB-51 RGB
Single Frame	72.2	72.2	40.1
LSTM classifier	74.5	74.3	42.8
Composite LSTM Model + Finetuning	**75.8**	**74.9**	**44.1**

Table 1. Summary of Results on Action Recognition.

Model	Cross Entropy on MNIST	Squared loss on image patches
Future Predictor	350.2	225.2
Composite Model	344.9	210.7
Conditional Future Predictor	343.5	221.3
Composite Model with Conditional Future Predictor	341.2	208.1

Table 2. Future prediction results on MNIST and image patches. All models use 2 layers of LSTMs.

We further ran similar experiments on the optical flow percepts extracted from the UCF-101 dataset. A temporal stream convolutional net, similar to the one proposed by Simonyan & Zisserman (2014b), was trained on single frame optical flows as well as on stacks of 10 optical flows. This gave an accuracy of 72.2% and 77.5% respectively. Here again, our models took 16 frames as input, reconstructed them and predicted 13 frames into the future. LSTMs with 128 hidden units improved the accuracy by 2.1% to 74.3% for the single frame case. Bigger LSTMs did not improve results. By pretraining the LSTM, we were able to further improve the classification to 74.9% ($\pm$0.1). For stacks of 10 frames we improved very slightly to 77.7%. These results are summarized in Table 1.

3.5. Comparison of Different Model Variants

The aim of this set of experiments is to compare the different variants of the model proposed in this paper. Since it is always possible to get lower reconstruction error by copying the inputs, we cannot use input reconstruction error as a measure of how well a model is doing. However, we can use the error in predicting the future as a reasonable measure of performance. We can also use the performance on supervised tasks as a proxy for how well the unsupervised model is doing. In this section, we present results from these two analyses.

Future prediction results are summarized in Table 2. For MNIST we compute the cross entropy of the predictions with respect to the ground truth. For natural image patches, we compute the squared loss. We see that the Composite Model always does a better job of predicting the future compared to the Future Predictor. This indicates that having the autoencoder along with the future predictor to force the model to remember more about the inputs actually helps predict the future better. Next, we compare each model with its conditional variant. Here, we find that the conditional models perform slightly better, as was also noted in Fig. 4.

The performance on action recognition achieved by finetuning different unsupervised learning models is summarized in Table 3. Besides running the experiments on the full UCF-101 and HMDB-51 datasets, we also ran the experiments on small subsets of these datasets where the effects of pretraining would be more pronounced. We find that all unsupervised models improve over the baseline LSTM which is itself well-regularized using dropout. The Autoencoder model seems to perform consistently better than the Future Predictor. The Composite model, which combines the two, does better than either one alone. Conditioning on the generated inputs does not seem to give a clear advantage over not doing so. The Composite Model with a conditional future predictor works the best, although its performance is almost same as that of the Composite Model without conditioning.

Method	UCF-101 small	UCF-101	HMDB-51 small	HMDB-51
Baseline LSTM	63.7	74.5	25.3	42.8
Autoencoder	66.2	75.1	28.6	44.0
Future Predictor	64.9	74.9	27.3	43.1
Conditional Autoencoder	65.8	74.8	27.9	43.1
Conditional Future Predictor	65.1	74.9	27.4	43.4
Composite Model	67.0	**75.8**	29.1	**44.1**
Composite Model with Conditional Future Predictor	**67.1**	**75.8**	**29.2**	44.0

Table 3. Comparison of different unsupervised pretraining methods. UCF-101 small is a subset containing 10 videos per class. HMDB-51 small contains 4 videos per class.

3.6. Comparison with Action Recognition Benchmarks

Finally, we compare our models to the state-of-the-art action recognition results. The performance is summarized in Table 4. The table is divided into three sets. The first set compares models that use only RGB data (single or multiple frames). The second set compares models that use explicitly computed flow features only. Models in the third set use both.

On RGB data, our model performs at par with the best deep models. It performs 4.7% better than the LRCN model that also used LSTMs on top of conv net features[1]. Our model performs better than C3D features that use a 3D convolutional net. However, when the C3D features are concatenated with fc6 percepts, they do slightly better than our model.

The improvement for flow features over using a randomly initialized LSTM network is quite small. We believe this is partly due to the fact that the flow percepts already capture a lot of the motion information that the LSTM would otherwise discover. Another contributing factor is that the temporal stream convolutional net that is used to extract flow percepts overfits very readily (in the sense that it gets almost zero training error but much higher test error) in spite of strong regularization. Therefore the statistics of the percepts might be different between the training and test sets. This is not the case for RGB percepts because the network there was trained on an entirely different dataset (ImageNet).

When we combine predictions from the RGB and flow models, we obtain 84.3 % accuracy on UCF-101. We believe further improvements can be made by running the model over different patch locations and mirroring the patches. Also, our model can be applied deeper inside the conv net instead of just at the top-level.

Method	UCF-101	HMDB-51
Spatial Convolutional Net (Simonyan & Zisserman, 2014a)	73.0	40.5
C3D (Tran et al., 2014)	72.3	-
C3D + fc6 (Tran et al., 2014)	**76.4**	-
LRCN (Donahue et al., 2014)	71.1	-
Composite LSTM Model	75.8	44.0
Temporal Convolutional Net (Simonyan & Zisserman, 2014a)	**83.7**	54.6
LRCN (Donahue et al., 2014)	77.0	-
Composite LSTM Model	77.7	-
LRCN (Donahue et al., 2014)	82.9	-
Two-stream Convolutional Net (Simonyan & Zisserman, 2014a)	88.0	59.4
Multi-skip feature stacking (Lan et al., 2014)	**89.1**	**65.1**
Composite LSTM Model	84.3	-

Table 4. Comparison with state-of-the-art action recognition models.

4. Conclusions

We proposed models based on LSTMs that can learn good video representations. We compared them and analyzed their properties through visualizations. More detailed analysis can be found in the expanded version of this paper (Srivastava et al., 2015). There we found that on the moving MNIST digits dataset, the model was able to generate persistent motion over long periods of time into the future even though it was trained for much shorter time scales. The learned features at the encoder and decoder when visualized show some important qualitative differences. In terms of performance on supervised tasks, we managed to get modest improvements only. The best performing model was the Composite Model that combined an autoencoder and a future predictor. The conditional variants did not give any significant improvements in terms of classification accuracy after finetuning, however they did give slightly lower prediction errors. More powerful decoders which incorporate some form of stochasticity are required to further address this question.

To get improvements on supervised tasks, the model can be extended by applying it convolutionally across patches of the video and stacking multiple layers of such models. In our future work, we plan to build temporal models from the bottom up instead of using them only to model high-level percepts. We will also use more powerful decoders that can model multimodal target distributions.

[1]However, the improvement is only partially from unsupervised learning, since we used a better conv net model.

Acknowledgments

We gratefully acknowledge support from IARPA ALADDIN project, ONR Grant N00014-14-1-0232, Samsung, and NVIDIA Corporation with the donation of a GPU used for this research.

References

Cho, Kyunghyun, van Merrienboer, Bart, Gülçehre, Çaglar, Bahdanau, Dzmitry, Bougares, Fethi, Schwenk, Holger, and Bengio, Yoshua. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*, pp. 1724–1734, 2014.

Donahue, Jeff, Hendricks, Lisa Anne, Guadarrama, Sergio, Rohrbach, Marcus, Venugopalan, Subhashini, Saenko, Kate, and Darrell, Trevor. Long-term recurrent convolutional networks for visual recognition and description. *CoRR*, abs/1411.4389, 2014.

Graves, Alex. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013.

Graves, Alex and Jaitly, Navdeep. Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1764–1772, 2014.

Gregor, Karol, Danihelka, Ivo, Graves, Alex, and Wierstra, Daan. DRAW: A recurrent neural network for image generation. *CoRR*, abs/1502.04623, 2015.

Hurri, Jarmo and Hyvärinen, Aapo. Simple-cell-like receptive fields maximize temporal coherence in natural video. *Neural Computation*, 15(3):663–691, 2003.

Ji, Shuiwang, Xu, Wei, Yang, Ming, and Yu, Kai. 3d convolutional neural networks for human action recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(1):221–231, Jan 2013.

Karpathy, Andrej, Toderici, George, Shetty, Sanketh, Leung, Thomas, Sukthankar, Rahul, and Fei-Fei, Li. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

Kingma, Diederik P. and Welling, Max. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., and Serre, T. HMDB: a large video database for human motion recognition. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

Lan, Zhen-Zhong, Lin, Ming, Li, Xuanchong, Hauptmann, Alexander G., and Raj, Bhiksha. Beyond gaussian pyramid: Multi-skip feature stacking for action recognition. *CoRR*, abs/1411.6660, 2014.

Le, Q. V., Zou, W., Yeung, S. Y., and Ng, A. Y. Learning hierarchical spatio-temporal features for action recognition with independent subspace analysis. In *CVPR*, 2011.

Memisevic, Roland. Learning to relate images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1829–1846, 2013.

Memisevic, Roland and Hinton, Geoffrey E. Learning to represent spatial transformations with factored higher-order boltzmann machines. *Neural Computation*, 22(6): 1473–1492, June 2010.

Michalski, Vincent, Memisevic, Roland, and Konda, Kishore. Modeling deep temporal dependencies with recurrent grammar cells. In *Advances in Neural Information Processing Systems 27*, pp. 1925–1933. Curran Associates, Inc., 2014.

Mobahi, Hossein, Collobert, Ronan, and Weston, Jason. Deep learning from temporal coherence in video. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 737–744, New York, NY, USA, 2009. ACM.

Ranzato, Marc'Aurelio, Szlam, Arthur, Bruna, Joan, Mathieu, Michaël, Collobert, Ronan, and Chopra, Sumit. Video (language) modeling: a baseline for generative models of natural videos. *CoRR*, abs/1412.6604, 2014.

Simonyan, K. and Zisserman, A. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, 2014a.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014b.

Soomro, K., Roshan Zamir, A., and Shah, M. UCF101: A dataset of 101 human actions classes from videos in the wild. In *CRCV-TR-12-01*, 2012.

Srivastava, Nitish, Mansimov, Elman, and Salakhutdinov, Ruslan. Unsupervised learning of video representations using LSTMs. *CoRR*, abs/1502.04681, 2015.

Susskind, J., Memisevic, R., Hinton, G., and Pollefeys, M. Modeling the joint density of two images under a variety of transformations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc V. V. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pp. 3104–3112. 2014.

Tran, Du, Bourdev, Lubomir D., Fergus, Rob, Torresani, Lorenzo, and Paluri, Manohar. C3D: generic features for video analysis. *CoRR*, abs/1412.0767, 2014.

van Hateren, J. H. and Ruderman, D. L. Independent component analysis of natural image sequences yields spatio-temporal filters similar to simple cells in primary visual cortex. *Proceedings. Biological sciences / The Royal Society*, 265(1412):2315–2320, 1998.

Vinyals, Oriol, Toshev, Alexander, Bengio, Samy, and Erhan, Dumitru. Show and tell: A neural image caption generator. *CoRR*, abs/1411.4555, 2014.

Wang, Zhou, Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. Image quality assessment: From error visibility to structural similarity. *Trans. Img. Proc.*, 13(4):600–612, 2004.

Zaremba, Wojciech, Sutskever, Ilya, and Vinyals, Oriol. Recurrent neural network regularization. *CoRR*, abs/1409.2329, 2014.