Reproducing Action Recognition Experiments with Modern Classification Networks

Dewal Gupta
UC San Diego
degupta@ucsd.edu

Mithun Dharmaraj
UC San Diego
mdharmar@ucsd.edu

Patrick Hayes
UC San Diego
phhayes@ucsd.edu

Abstract

In this paper we reproduce various experiments for trimmed action recognition using modern deep neural-networks. As the state-of-the-art deep learning architectures continue to improve we wanted to demonstrate that performance on action recognition can be improved by using more modern image classification networks as a backbone architecture. For a baseline, we train an GoogLeNet model with S3D convolutions from scratch on our on mini-kinetics dataset with 5 categories. An S3D convolution uses separable convolutions to decouple the spatial and temporal features in video. We call this baseline model S3D-v1. We then update our baseline model to use a more modern inception-v3 style architecture, but still use S3D convolutions. We call this model S3D-v3. We expected that the S3D-v3 would perform better than the S3D-v1 because the inception-v3 performs better than the inception-v1 on image classification, but empirically we found that S3D-v1 performed better. These unexpected results may be due to the limited training data we used for training. We would expect the larger S3D-v3 to overfit more to a smaller dataset. It may also be more difficult to take more hyper parameter tuning to train the deeper S3D-v3.

1. Introduction

Action Recognition, like many other computer vision tasks, has seen dramatic improvement to the state-of-the-art with the application of large labeled datasets and deep convolutional neural networks (CNNs). While there is an ever increasing library of action recognition datasets available, many top performing action recognition models still rely on the size and variety of ImageNet to pretrain their networks. Furthermore, many action recognition models are largely based on top performing image classification models. At an extreme, action recognition models treat a video as a collection of individual images and average the results of classifying each frame of a video. However this simplification throws out the temporal features of a video. Finding a suitable architecture to extract these temporal features has been the objective of numerous recent experiments. In this paper we trained from scratch a model that incorporates the findings of many of these recent experiments.

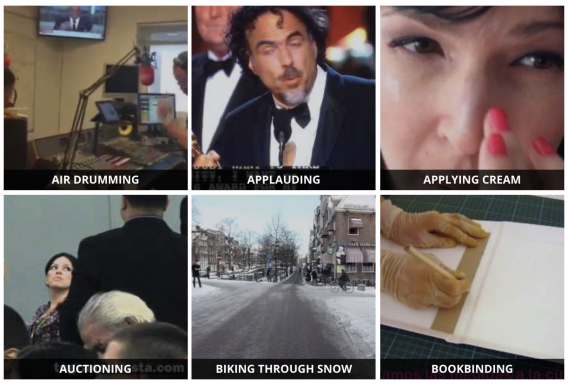

Figure 1. A view of the Kinetics Action Recognition classes

The temporal features captured in videos can be used in a wide variety a classification task. A simple example would be distinguishing between a door opening or closing, but temporal features could also be used to distinguish between a golf swing with proper or poor form. A smart surveillance system could use temporal features to identify suspicious behavior, or a smart baby monitor could use temporal features to decide when to record home videos.

To encourage progress in the field of action recognition the ActivityNet challenge was introduced. The ActivityNet challenge formalized a set of action recognition tasks by providing large diverse datasets to train and evaluate different models. The tasks include trimmed and untrimmed video classification, temporal action proposals, action localization, and dense captioning events in videos. In this paper we focus on the trimmed video classification

challenge. The trimmed video classification challenge involves classifying approximately 10 second videos across 400 human action categories. The categories include a wide variety of human actions such as applauding, jogging, and kayaking. Each category contains at least 400 video clips for each category. The top performing models for the 2017 challenge achieved an average error rate of 12.39%. Where average error rate is the average of the top 1 error rate and the top 5 error rate.

We built two models to evaluate on the trimmed video challenge. We trained from scratch the S3D-v1 model that was proposed in [5] and we modified the S3D-v1 model to use inception-v3 as its backbone architecture. We evaluated these two models on the trimmed video classification challenge to analyses how updating the backbone architecture to a modern image classification network would affect performance. We found that updating the backbone architecture to a modern image classification network did not lead to the same kind of accuracy improvements that were seen in image classification, but these results may caused by difficulties in training.

2. Related Work

Many models for computer vision tasks, from semantic segmentation to optical flow to object detection have used image classification architectures as a backbone for their model. A major reason for this is that image classification requires a network to build accurate and representative feature maps that are mainly task independent. This has allowed researchers to transfer these networks to tasks like semantic segmentation, object detection, and depth estimation. Another major advantage is that ImageNet architectures provide researchers a way to pre-train their deep models on the large ImageNet dataset to help speed up training and prevent over-fitting.

This has been largely true in computer vision as well, where most successful architectures have come from ImageNet. Zisserman et al. showed that I3D, a network based on the inception v1 architecture, works amazingly well with few modifications to the base architecture [1]. Other researchers have been successful at modifying other architectures such as ResNets [2] to decent results as well.

The biggest challenge that remains is creating deeper networks while mitigating the impact of computational complexity. This is largely due to the use of expensive 3D convolutions. Researchers have looked at other methods such as using attention and other mechanisms for dealing with the temporal domain but in this work we restrict our experiments to 3D convolutions. Other works have focused on mitigating the cost of using 3D convolutions, and one idea that has stuck out has been the use of separable convolutions [5]. Xie et al. were able to build on top of I3D where they took the expensive 3D convolutions and broken them into 2 convolutions: a spatial convolution followed by a temporal one. This helps reduce 30% of the parameters from the I3D model and it actually performs slightly better as well.

The other challenge is that despite using 3D convolutions, the networks are not able to fully learn the motion between frames and therefore arent fully learning the features necessary for action detection. Zisserman et al. experimented with two stream models where they utilised optical flow as another type of input (instead of RGB frames) to the network. They are able to ensemble the two (RGB and optical flow) networks and show a significant improvement in performance. Though this is not ideal, we do not conduct any further experiments nor do we use optical flow in our study. In the future, it would be best if a single network was capable of capturing motion and using it to classify actions. It might be worthwhile in the future to examine the impact of using leading optical flow networks for classification rather than ImageNet architectures. The state-of-the-art models for image classification are constantly improving. In the years after Inception-v1 was introduced inception versions 2, 3, and 4 were also introduced along with Inception-ResNet-v1 and 2. These iterations incrementally improved image classification accuracy by introducing batch normalization, modifying the kernel sizes, adding skip connections, and residual connections.

In this work we examine the feasibility of using deeper networks such as Inception v3 and using these architectures for video classification. These networks have shown a much lower error on ImageNet competitions and might be able to also outperform on the Kinetics Dataset.

3. Architecture

Our baseline model, is based of the model proposed in [5]. The model uses GoogLeNet as a backbone architecture. The convolutions are inflated to account for the third temporal dimension. We believe that 3D convolution is well-suited for spatio temporal feature learning. Our model has the ability to model temporal information better owing to 3D convolution and 3D pooling operations. Figure 3 shows that a 2D convolution applied to 3D input, outputs an image. We lose all temporal information from the input. Only 3D convolution preserves the temporal information of the input signals resulting in an output volume. The same phenomenon is applicable for 3D pooling.

One downside of the 3D convolution is the exponential increase in the number of parameters during inflation from

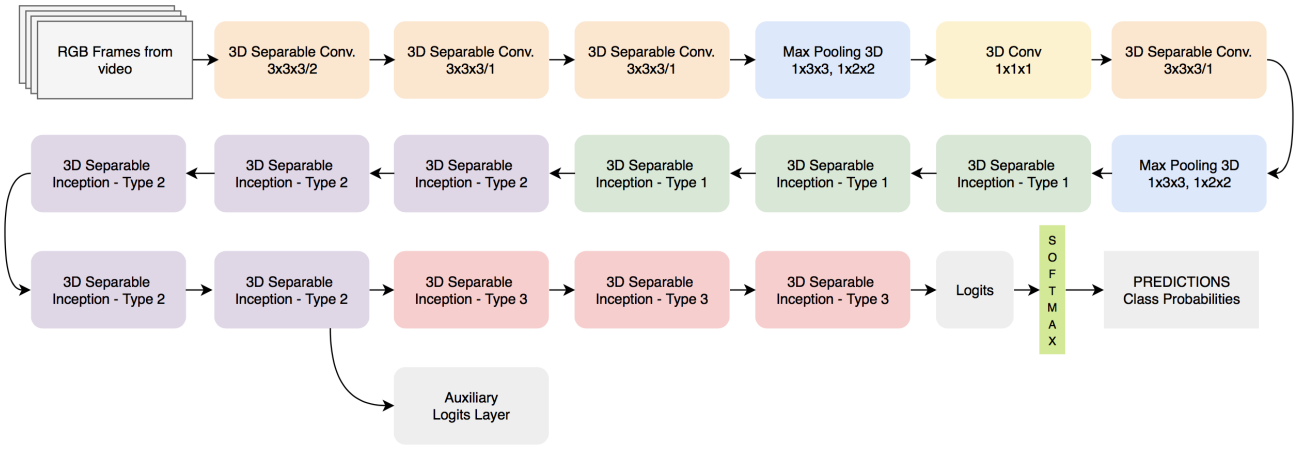

Figure 2. S3D-v3 Model Architecture

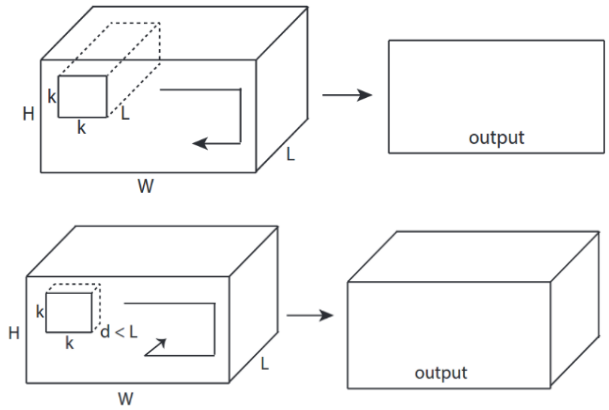

Figure 3. Figure on top illustrates 2D convolution applied to 3D data. The bottom one illustrates a simple 3D convolution resulting in an output volume. These figures are taken from [4]

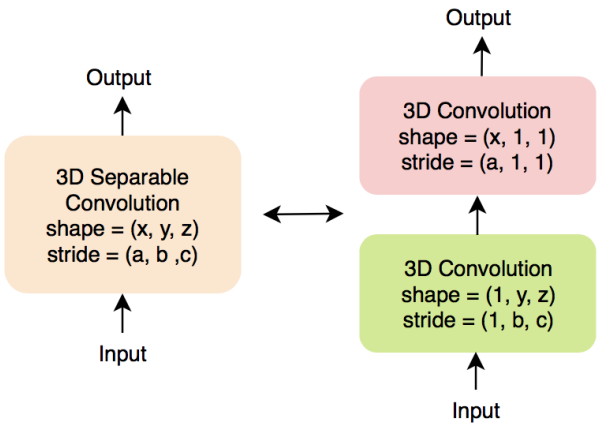

Figure 4. Seperable 3D convolutions

2D to 3D. To mitigate this spike in parameters, we adapt an idea discussed in [5]. We replace the 3D convolutions with spatiotemporal-separable 3D convolutions as shown in Figure 4, A separable convolution is 1.5x more computationally efficient than the conventional 3D convolutions.

Our model, shown in Figure 2, built on an Inception v3 backbone inflates the 2D convolutions to 3D Separable Convolutions. Following this all the inception modules in Inception v3 model are inflated to Separable convolutions. Note that we have a unit 3D convolutional layer which typically modifies the output dimensions of the network. Unit convolutions [1] in between a series of 3D convolutions can speed up convolution by reducing the dimension. Table 1 lists all the layers in our model, along with the type, kernel shape and stride. The input size specifies the shape of the data inputs to that layer. The first dimension specifies the temporal dimension, while the next two specifies the spatial dimension. The last dimension is the number of input channels for that layer. The input channel for the next layer would directly be equal to the output channel for the current layer. The three different types of inception modules are shown in Figures 5, 6, 7

[3] introduced auxiliary classifiers to combat the effect of vanishing gradient in deep networks. The auxiliary logit layer in our model works by pushing the useful gradients to the lower layers to make them immediately useful and improve the convergence during training. However, [3] argues that, auxiliary classifier does not help the early stages of training. The training progression of a network with and without the layer virtually looks the same. However, once

Table 1. S3D-v3 Architecture details

Type	Kernel shape and Stride	Input Size
3D Sep. Conv	3x3x3 / 2	64x224x224x3
3D Sep. Conv	3x3x3 / 1	32x112x112x32
3D Sep. Conv	3x3x3 / 1	32x112x112x32
Max Pool	3x3x3 / 2	32x112x112x64
3D Conv	1x1x1	16x56x56x64
3D Sep. Conv	3x3x3	16x56x56x80
Max Pool	3x3x3 / 2	16x56x56x192
Inception-v3 Type 1	see Fig	8x28x28x192
Inception-v3 Type 2	see Fig	3x13x13x768
Inception-v3 Type 3	see Fig	1x6x6x2048
Linear Avg Pool	Logits	1x6x6x2048

higher accuracy is reached, the network with the auxiliary logits layer overtakes the other and plateaus at a much lower error rate. The classifier logits layer including the auxiliary logits are batch normalized or has a dropout layer to act as a regularizer. The resulting model including all the inception modules is 42 layers deep and has twice as many parameters as the GoogLeNet based architecture.

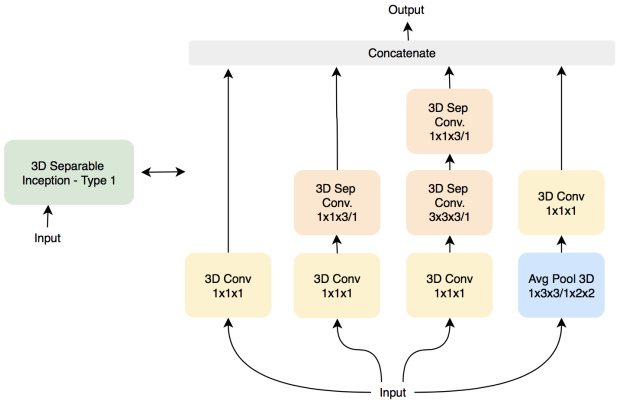

Figure 5. Expanded Type 1 Inception blocks

4. Experiments

First, we replicated the results from [1] and [5] to establish baselines with which we could compare our models. Due to limited time and computational resources we did not pre-train either network, and chose to train all networks from scratch. We tested our models using the kinetics dataset that contains about 300,000 videos spanning over 400 categories. However, again due to our limited resources, we randomly restricted the dataset to 5 categories and for each category we used roughly 400 videos for training and 25 for validation.

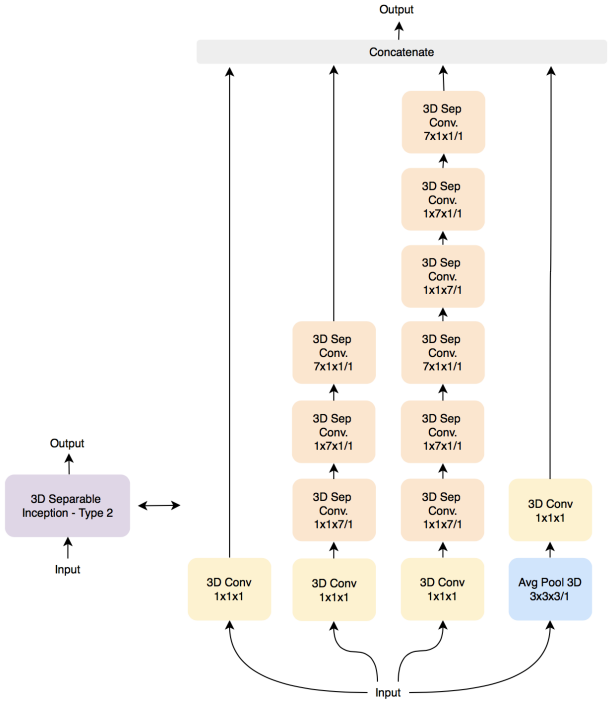

Figure 6. Expanded Type 2 Inception blocks

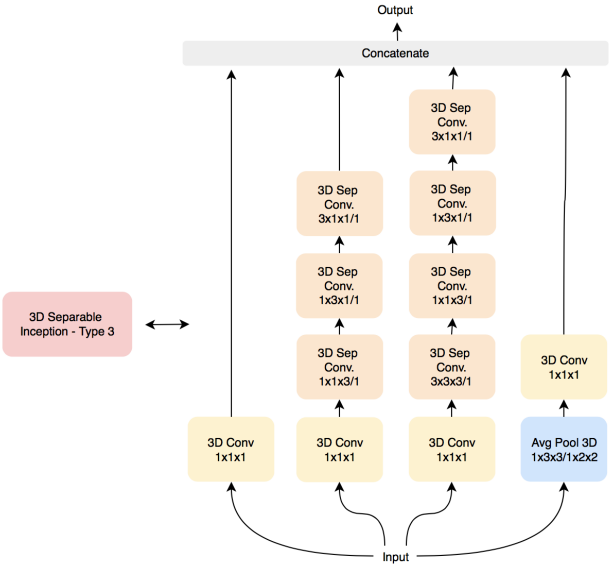

Figure 7. Expanded Type 3 Inception blocks

For training, we used two NVIDIA GTX1080Ti GPUs on the UCSD Data Science Cluster, and trained all networks for 24 hours. We did not use any adaptive learning rate but manually decreased the learning rate from 0.01 to 0.001 after 12 hours. Each training iteration was batched with

5 videos, and for every epoch, not only were the videos shuffled, but also the frames in those videos (following the procedure established by Zisserman et al.). We also used stochastic gradient descent along with a cross-entropy loss function for all network training.

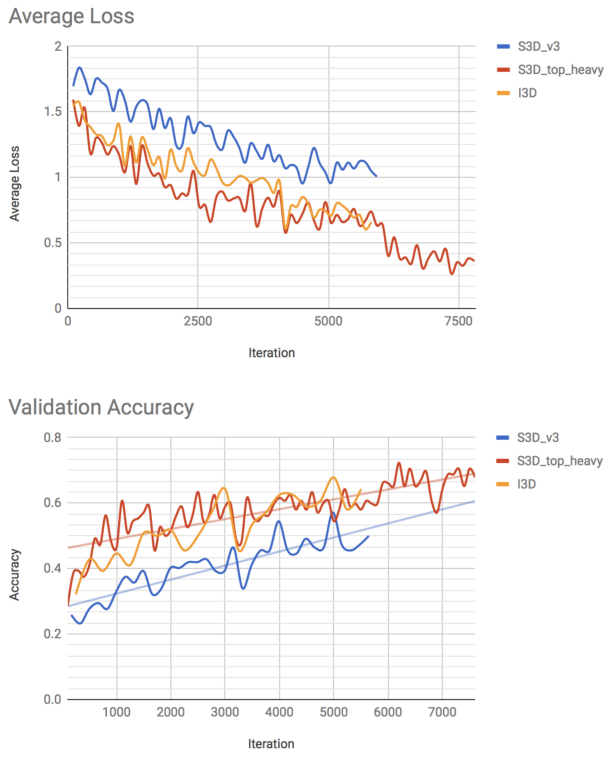

Figure 8. Loss and Accuracy on Validation

The I3D and S3D models performed well as expected: S3D ended up with a top-1 accuracy of roughly 68% whereas the I3D performed at a slightly lower accuracy, around 64%. This was expected as the S3D published results show it performing marginally better than the I3D though both works pre-train the networks and utilise the full kinetics dataset. We also trained and tested our Inception V3 with Separable convolutions (S3D-v3). As seen in the figure, the S3D-v3 model did not perform as well as the other two, consistently performing at a roughly 5-10% lower accuracy.

5. Analysis

Contrary to what we expected, the S3D network with inception v3 backbone has higher error rate when compared to the one with GoogLeNet architecture [1]. This may be because of the Inception v3 architecture itself which has almost twice as many parameters as GoogLeNet. Since were training the network from scratch, we attribute the high error rate to the limited amount of data we had. Given more data, it is quite possible that the inception v3 model would have outperformed the original v1 based models.

To confirm this, we made certain key modifications to the S3D model with inception v3 architecture, to reduce the amount of parameters - almost to half the original size. Essentially we only spatially convoluted (2D convolutions) for the first half of the network. All temporal convolutions were carried out after layer 9. We call this a Top-Heavy model - signifying which part of the network carries out the temporal convolutions. The top-heavy model performs on-par with the network running on GoogLeNet architecture. Even though we see a better increasing trend with Inception v3, the networks with less number of parameters trains faster for the MiniKinetics dataset.

Lastly, another possible reason could be the convolutional layers we added for the temporal domain. Instead of simply inflating the 2D convolutions to 3D ones, we instead added extra layers for the temporal domain while keeping the original convolutions. One example is in Figure 6 where there are repeated 1x7 and 7x1 convolutions in the original network. We chose to make this into 3 convolutional layers: 1x1x7, 1x7x1, and 7x1x1. In the future, we would love to see how the network would do by simply inflated the original two layers into 7x1x7 and 7x7x1. This would allow the network to potentially better understanding temporal changes correlated along the x or y axis. Right now our model is doing a 7x1x1 convolution which may inhibit it from learning motion along the x or y direction.

6. Conclusion

In this work we found inconclusive results. Adapting an action recognition model to use a modern deep learning architecture may or may not improve your accuracies depending on your training resources and the deep learning architecture you are updating too. Updating from Inception v1 to Inception v3 requires additional training time and a larger minimum training dataset. If you have those resources we would expect a action recognition model base on inception-v3 would outperform a model based on inception-v1. Deep learning architecture that are even more modern than inception-v3 may have different training requirements and may interact differently with the S3D convolutions. As deep learning networks continue to evolve we expect the model selection process to grow with the variation in architectures. Without stronger theories behind what makes a model perform well it is difficult to predict what architectures will perform the best without evaluating all of the possible options. A streamlined system for evaluating pro-

totype models would dramatically speed up the model selection process and improve accuracies on deep learning applications.

7. Future Work

To achieve more conclusive results our experiments should be repeated on the entirety of the Kinetics dataset rather than just five categories. Further work would also include experimenting with more network architectures. Inception-v3 is only one of many advances in deep learning architectures experimenting with ResNet, DenseNet, Squeeze-and-Excitation networks, and dilated filters would all be valuable future work. Until deep learning matures as theoretical field, the model selection process will involve time consuming empirical trials. Valuable future work would be developing systems level support for conducting these empirical trials to reduce the amount of work required to compare two models.

References

[1] J. Carreira and A. Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. *CoRR*, abs/1705.07750, 2017.

[2] G. Hinton, O. Vinyals, and J. Dean. Distilling the Knowledge in a Neural Network. *ArXiv e-prints*, Mar. 2015.

[3] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[4] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri. C3D: generic features for video analysis. *CoRR*, abs/1412.0767, 2014.

[5] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy. Rethinking Spatiotemporal Feature Learning For Video Understanding. *ArXiv e-prints*, Dec. 2017.