# A Baseline Comparison for Action Understanding with Multiple Classes of Actors

Zhongkai Shangguan, Yue Zhao, Jingwen Wang
University of Rochester
zshangg2, yzhao88, jwang191 @ur.rochester.edu

## Abstract

*ResNet architectures have shown strong performance for multiple-label classification. In this paper we use actor-action dataset (A2D) for action recognition in video dimension. Based on 2D ResNet, we apply the SE (Squeeze-and-Excitation) blocks to the ResNeXt architecture which can efficiently exploit the split-transform-merge strategy and learn feature of different channels. We implement and compare several architectures: Resnet34, PNASNet-5-large, EfficientNet-B7 and SE-ResNeXt101 to this training data set and our results shows that SE-ResNeXt101 have the best performance.*

## 1. Introduction

In order to represent simple action combination, most actions recognition task rely on handcrafted features, but those are gradually shown to be efficiently replaced by Convolutional Neural Networks (CNN). Many image analysis based pipelines are given, which take raw image as the input and output the classification labels of each image [6]. Several CNN architectures have shown state-of-the-arts performance on object recognition [8] [10]. It is comparably challenging when expand the 2D input to a state-of-the-art 3D dimension video data, and one of the most popular method is to use stacked video frames. For actor-action dataset (A2D) [11], capturing different actors and their actions consist of segmentation and multi-classification and in this paper we focus on the multi-classification scenario. A2D contains 3782 videos from YouTube, in each of which, objects are annotated with actor-action label. There are overall 43 valid actor-action tuple which is formed by seven actor classes (adult, baby, ball, bird, car, cat, and dog) and eight action classes (climb, crawl, eat, fly, jump, roll, run, and walk) not including the no-action class.

ResNet [4] (Residual CNNs) is one of the most commonly used architectures or backbones for multi-label tasks, as it can efficiently solve the degradation problem when increasing the accuracy lead to deeper CNN layers. 3D ResNet or spacial ResNet is more commonly used in solving the action recognition problem of the video [5] [9]. However,3D CNNs heavily rely on the modification of the architecture. In this paper, based on the 2D ResNet architecture, we implemented SE-ResNext101 on A2D and compare with the related architectures, such as EfficientNet-B7.

Our paper introduce the overall implementation with data augmentation, network architecture and optimization methods in Sec. 2, and the experiment of different architectures is shown in Sec. 3. The result and conclusion is shown in Sec.4. Our pre-trained parameters are initialized by ImageNet.

## 2. Method

This section provides details of the training method, including data augmentation, model structure, loss function and optimization method.

### 2.1. Data Augmentation

By doing image augmentation, people can get a better performance when dealing with limited image dataset and avoid over-fit, this will improve the model robustness both on validation and test set.

Some normal augmentation methods like rotation and flip are applied in this project. Before feeding the data to model network, we crop, pad and re-scale the images into (224, 224, 3).

As the images are extracted from videos, motion blur should also be considered as an important part, this project also use Gaussian Blur, Median Blur and Motion Blur for pre-processing images.

Dropout is another powerful technique which aim to make the images more similar to our daily environment. This technique will randomly generate less than five $8 \times 8$ black squares in order to simulate occlusion of objects in real scenes.

Apart from the above methods, we also modify image contrast by using RandomGamma, HueSaturationValue

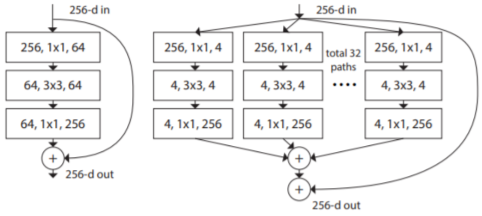

Figure 1. Squeeze-and-Excitation (SE) Block

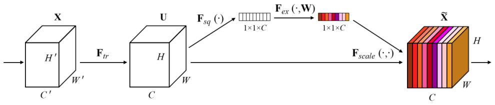

Figure 2. Squeeze-and-Excitation (SE) Block

change, and RandomBrightnessContrast, all these augmentation methods are based on augmentation library [1].

### 2.2. Network Architecture

We modified and test different CNN architectures, including ResNet34, PNASNet-5-large, Efficientnet-b7 and SE-ResNeXt101. All models are initialized from pretrained parameters on ImageNet [2]. Among all this models, SE-ResNeXt101 shows the best performance.

ResNeXt is a simple architecture which adopts VGG/ResNets' strategy of repeating layers, while exploiting the split-transform-merge strategy in an easy, extensible way. The innovation lies in the proposed aggregate transformations, which replaces the original ResNet's three-layer convolution block with a parallel stack of blocks of the same topology structure (Figure 1), which improves the accuracy of the model without significantly increasing the magnitude of the parameter.

Then apply SE (Squeeze-and-Excitation) blocks to ResNext we get SE-ResNext model, the SE block is shown in (Figure 2), the main idea of SE block is by giving different channels a weight which make the model also learn feature of channels. We notice that after convolution, we first apply global maxpooling to squeeze the 2-d channel into a number, then using fully-connected layers learn the weight of each channel, finally multiply the weights to the original 2-d feature.

Finally add a dense layer as a classifier after the Feature extractor. This layer use Mish[7] as the activation function, and it is followed by a Dropout layer in order to drop redundant features and improve the model robustness.

### 2.3. Loss and Optimization

The loss function is defined as Binary Cross Entropy.

$$Loss = -w_n[y_n \cdot log\sigma(x_n) + (1-y_n) \cdot log(1-\sigma(x_n))] \quad (1)$$

The optimization method is SGD (Stochastic Gradient Descent) with two stages, in stage one CosineAnnealingLR is applied with a large initial learning rate, which can help accelerate the converge process. The second stage is used to fun-tune the model in a more meticulous way, with a circle scheduler start with a very small learning rate. Additionally, the training process is initialized with amp(AUTOMATIC MIXED PRECISION), which enable a Mixed Precision training process. It contains two computing type: FP16 (Half-precision 16-bit floating-point) and FP32 (Single-precision 32-bit floating-point).

## 3. Experiment

Two local 2080Ti with 11GB memory were used for the hardware, the Ubuntu 16.04 and 18.04 were used for the system, and the Cuda 10.1 , Python 3.7, PyTorch 1.2.0 were used for the environment. For the performance in the experiment, the three information we collected are precision, recall and F1 score for the pattern recognition. To be specific, precision represents the positive predictive value, which is the fraction of relevant instances among the retrieved instance; recall represents sensitivity, which is the fraction of the total amount of relevant instances that were retrieved and it is the number of correct positive results divided by the number of all relevant samples; F1 score is the measure of a test's accuracy, which is computed from the precision and recall [3]. The Fig.3 shows the training loss and training accuracy of SE-ResNeXt101 with the training hyperparameters Batch size =64, Epoch=200, learning rate =0.05, accumulate=1, Step =10. The left figure shows the training loss of SE-ResNeXt and we can find that the Training loss rapidly decreased to 0.1121 when epoch reached 2, and finally slowly converged to the certain value around 0.03-0.04. The right figure shows the training accuracy, and it has huge accuracy improvement from the epoch 1 to epoch 28, and then coverage slowly to the epoch 192 eventually. The sinusoidal shape has been observed for the training accuracy since the optimization method SGD with two stages were used and they are CosineAnnealingLR and MutiStepLR. The training Time for SE-ResNeXt101 is average 90s per epoch and the training loss is 0.1258 when it finishes the first epoch. The best performance at epoch 192 and the number of iterations is 3300.

We also did the comparison with different models such as Resnet34, PNANet 5 large, Efficient Net-B7 (with and without augmentation), SE-ResNeXt101(with and without augmentation), and the result in fig.4 shows that SE-ResNeXt101 with the activation function Mish has the best performance.

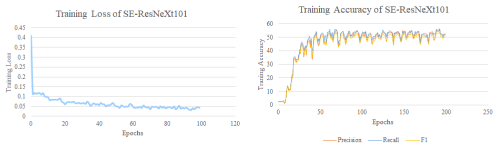

Figure 3. Training Loss and training accuracy of SE-ResNeXt101

| Model | Train Set | | | Validation Set | | |
|---|---|---|---|---|---|---|
| | Precision | Recall | F1 | Precision | Recall | F1 |
| Resnet34 | / | / | / | 48.4 | 46.0 | 45.8 |
| PNANet 5 large* | / | / | / | 51.3 | 54.3 | 50.7 |
| Efficientnet-B7 | 79.2 | 77.3 | 77.1 | 54.1 | 54.0 | 52.4 |
| SE-ResNeXt101 | 71.7 | 71.8 | 71.2 | 58.9 | 59.8 | 57.5 |
| Efficientnet-B7 (Mish Aug) | 78.9 | 75.7 | 76.0 | 56.3 | 58.9 | 56.1 |
| SE-ResNeXt101 (Mish Aug) | 84.8 | 82.8 | 83.0 | 60.6 | 61.1 | 59.2 |

(* indicate not finetune)

Figure 4. The results comparison with different models

## 4. Conclusion

SE-ResNeXt101 shows better performance for the validation set with the precision of 60.6, recall of 61.1 and F1 score of 59.2. Our experiments shows that the optimized model architecture contributes to the accuracy of the results. Proper data augmentation can also improve the performance,while other image pre-processing methods such as vertical flip, transpose and all sharpen methods (sharpen, CLAHE) are not useful. The smoothing training process methods, including optimizer selection, two stage for training using different scheduler and accumulator, also decrease the training time and contribute to higher accuracy. For future works, we can try new models and activation method as well as blend models. Also, combining two learning rate into one can be a good direction for new approaches to this task.

## References

[1] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. Albumentations: Fast and flexible image augmentations. *Information*, 16(1), 2020.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[3] C. Goutte and E. Gaussier. A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation. 2005.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[5] Hirokatsu Kataoka, Tenga Wakamiya, Kensho Hara, and Yutaka Satoh. Would mega-scale datasets further enhance spatiotemporal 3d cnns?, 2020.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017.

[7] Diganta Misra. Mish: A Self Regularized Non-Monotonic Neural Activation Function. *Information*, 2019.

[8] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks, 2013.

[9] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos, 2014.

[10] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[11] C. Xu, Shao-Hang Hsieh, C. Xiong, and J. J. Corso. Can humans fly? action understanding with multiple classes of actors. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2264–2273, 2015.