# **Figure Skating Jumps with Video Classification**

*Elena Golubeva, Vladislav Goteiner*

*Holon Institute of Technology, Faculty of Applied Mathematics 2020*

**Abstract**. Videos can be understood as a series of individual images; and therefore, many deep learning practitioners would be quick to treat video classification as performing image classification a total of *N* times, where *N* is the total number of frames in a video. Taking this into consideration, naive video classification method would be: loop over all frames in the video file; for each frame, pass the frame through the CNN (e.g. Resnet50 in our case, which is a convolutional neural network that is 50 layers deep); classify each frame individually and independently of each other; obtain the predictions from the CNN; compute the average of the last *K* predictions and choose the label with the largest corresponding probability; label the frame and write the output to disk. We demonstrate effectiveness of the proposed method with additional layers by applying it to the action recognition tasks on RGB videos. Experiments show that *amount of frames and the addition of dense layers* improves the results of the method.

**1 Introduction**

Recently, video classification takes a big part in our day-to-day life. The sheer volume of video data has motivated approaches to automatically categorizing video contents according to classes such as human activities and complex events. There are various methods that are used to classify video: CNN, RNN, 3D ConvNets, MLP (multilayer Perceptron), etc. In this project, we are showing how simple CNN modifications with additional layers can give us great results without object detection. One of the reasons why we decided to work with figure skating is because it is a very specific problem that needs its very specific solution.

There are a lot of examples of video classification out there, so why is our project so special? Indeed, there is a lot of research done and algorithms build for classifying videos, but one crucial thing wasn’t clarified - those are classification of videos with very bold distinctions. For example, classify the type of sport (racing, tennis, boxing, swimming, etc), object classification (cat, human, pen, building, etc). But what if you want to classify tennis technique, swimming style, cat action, or human mood? We get a more narrow and more specific problem, to which the previous solution won’t work.

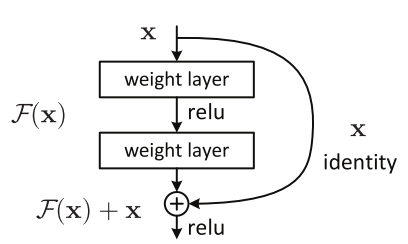
This is the type of problem we uncover in this project, where we try to classify one element in figure skating - jumps. We have a total of 7 jumps (classes): Axel, Euler, Flip, Loop, Lutz, Salchow, Toeloop. The differences between each of the jumps are in the beginning (what leg is on the ice - left or right, is skater starts the jump with the back or front) and in the end (on which foot skater lands).

**2 Related Work**

**2.1 ResNet**

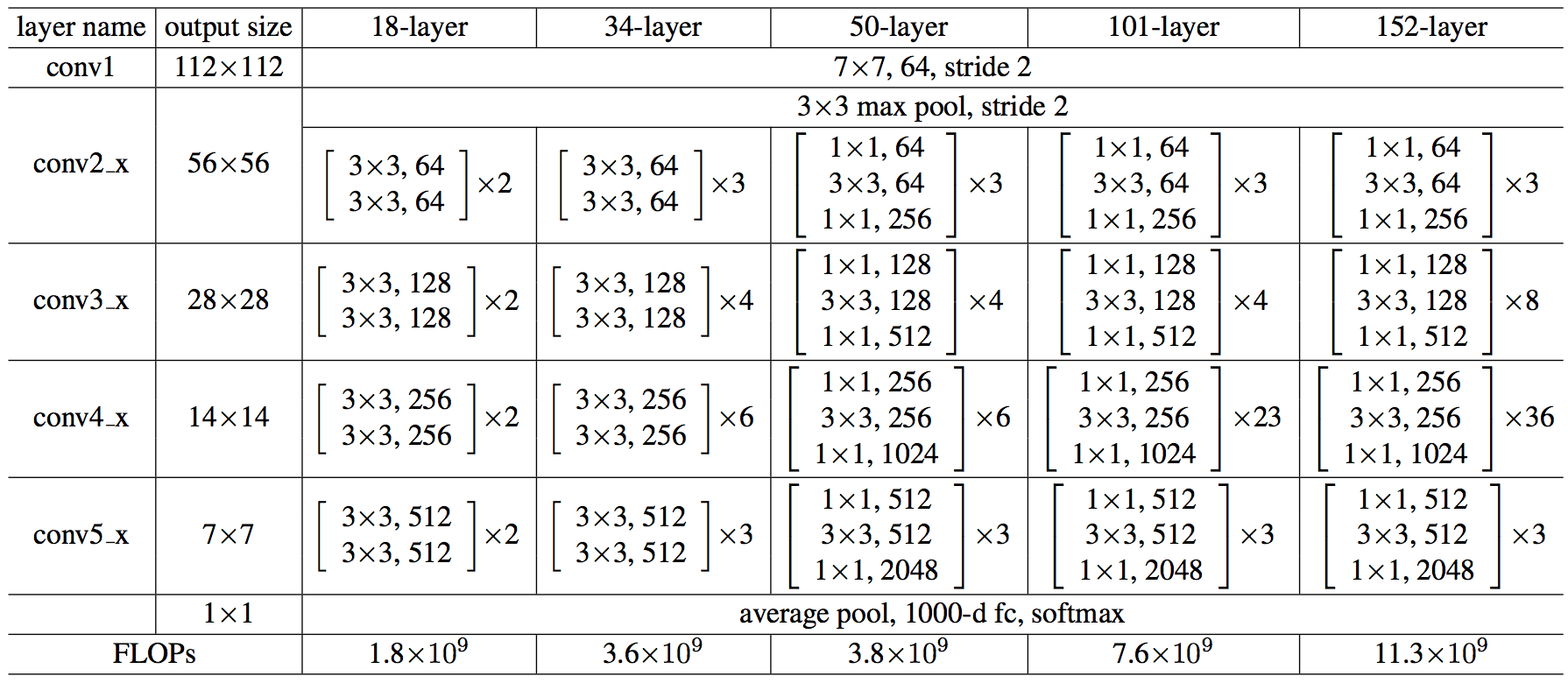
Residual CNN is used for image classification tasks. ResNet is great in solving the degradation problem (when adding more layers causes higher training error, because accuracy gets saturated).

Residual block:

**

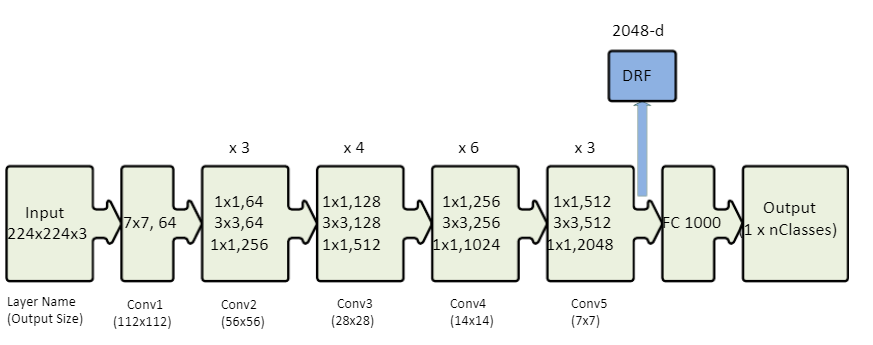
*Figure 1. Residual block in ResNet [1]*

The formulation of F(x)+x can be realized by feedforward neural networks with shortcut connections. The shortcut connections perform identity mapping, and their outputs are added to the outputs of the stacked layers. By using the residual network we can easily gain accuracy from greatly increased depth.



*Figure 2 [2]. Shown how different building blocks or bottlenecks make up different resnets. ResNet has a total of 5 sets of convolutions. The input size of the first group of convolutions is 224x224, and the output size of the fifth group of convolutions is 7x7, which is reduced by 32 (2^5) times. Each reduction is 2 times, a total of 5 reductions, and each time makes stride 2 on the first layer of each set of convolutions.*

ResNet50 is a 50 layer ResNet. Here each 2-layer block (each block in ResNet50 is 3 layers deep) is replaced in the 34-layer net with this 3-layer bottleneck block, resulting in a 50-layer ResNet (see table above). They use option 2 for increasing dimensions. This model has 3.8 billion FLOPs.



*Figure 3. ResNet-50 architecture [3] shown with the residual units, the size of the ﬁlters, and the outputs of each convolutional layer. DRF extracted from the last convolutional layer of this network is also shown. Key: The notation k×k, nin the convolutional layer block denotes a ﬁlter of size k and n channels.FC 1000 denotes the fully connected layer with 1000 neurons. The number on the top of the convolutional layer block represents the repetition of each unit. nClasses represents the number of output classes.*

**2.2 Datasets**

ImageNet is a dataset of millions of labeled high-resolution images belonging roughly to 22k categories. The images were collected from the internet and labeled by humans using a crowd-sourcing tool. Starting in 2010, as part of the Pascal Visual Object Challenge, an annual competition called the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC2013) has been held. ILSVRC uses a subset of ImageNet with roughly 1000 images in each of 1000 categories. There are approximately 1.2 million training images, 50k validation, and 150k testing images.

**2.3 Additional Layers**

ReLUs (Rectified Linear Units) are utilized to replace the tanh units, which makes the training process several times faster.

Dropout is introduced and has proven to be very effective in alleviating overfitting.

**3 Classification Algorithm**

As mentioned earlier, our algorithm [6] is as follows:

1. Loop over all frames in the video file and save them [5]
2. Pass each frame through the CNN
3. Gather the predictions from the CNN and maintain them in a list
4. Compute the average of the last K predictions and choose the largest corresponding probability
5. Label the frame and write the output frame to disk

There’s a little problem with this approach, when seeing the result (video) we encounter a sort of “prediction flickering”. This issue can be solved on a writing (5th) stage by frame frequency manipulation.

After intense research, we discovered that there are no existing datasets for a specific topic. We had to create our own dataset of classified videos. We did so by using ISU competition recordings: full-length competition video, from which we manually cut relevant snippets (jump execution) and organized them to the appropriate folder (each folder corresponds to specific class). One of our main obstacles was the lack of data. We found out that not all jumps are equally popular. Some jumps are being more popular or have longer execution (a class with more data), and some jumps are rare or very short (a class with less data). To ensure this doesn’t affect the algorithm accuracy we created an environment with an approximately equal amount of data - each class had the same number of videos.

Another challenge was the data [4] itself. As you can see below, not always we have a nice frame with defined objects, instead, sometimes, we have a frame where, even for a person, it’s hard to say what you are looking at:



Considering the fact that we go for multi-classification in the logistic regression model, we use softmax in the last layer instead of the sigmoid.

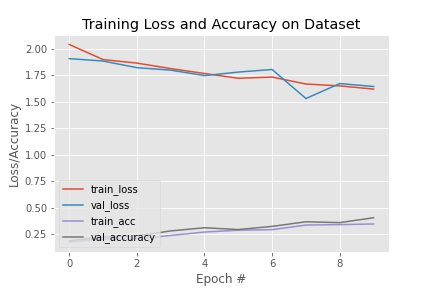
While experimenting we discovered, that not always a small learning rate of optimizer gives you great results. In our case we started with learning rate 0.001 and with 100 epochs couldn’t get an accuracy of over 41%. After increasing lr=0.01 and 10 epochs we finally saw expected results of >75%. We are not sure whether it is due to our data.

**4 Experiments**

We tried slightly different structures and also to optimize the parameters so that we will get the most optimal result. Here are some of the different structures we tried and what results we got with them:

structure:

1. ResNet50
2. AveragePooling2D
3. Flatten
4. Relu
5. Dropout 0.8
6. Relu
7. Dropout 0.5
8. Softmax

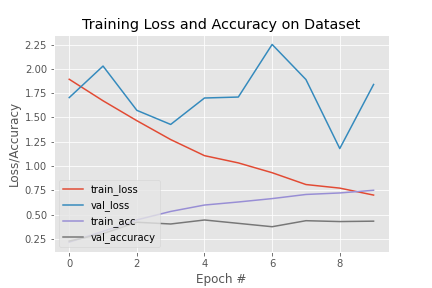
****

*Loss:1.5885 , Accuracy: 37%, Val loss: 1.6880, Val accuracy:0.3958*

*Learning rate: 0.001, Momentum: 0.9, Decay:0.01, Epochs: 10*

structure:

1. ResNet50
2. AveragePooling2D
3. Flatten
4. Relu
5. Sigmoid
6. Relu
7. Dropout 0.5
8. Softmax

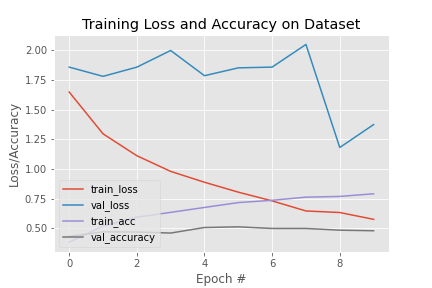
****

*Loss:0.7020 , Accuracy: 75%, Val loss: 1.8397, Val accuracy:0.4336*

*Learning rate: 0.01, Momentum: 0.9, Decay:0.001, Epochs: 10*

Structure:

1. ResNet50
2. AveragePooling2D
3. Flatten
4. Relu
5. Dropout 0.5
6. Softmax

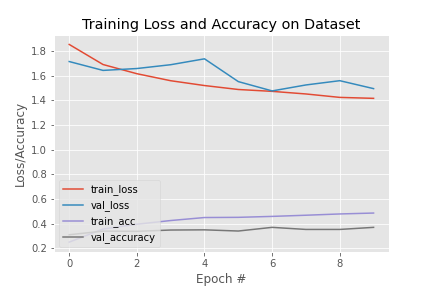
****

*Loss 0.5765 , Accuracy: 79%, Val loss: 1.3741, Val accuracy: 0.4803*

*Learning rate: 0.01, Momentum: 0.9, Decay:0.001, Epochs: 10*

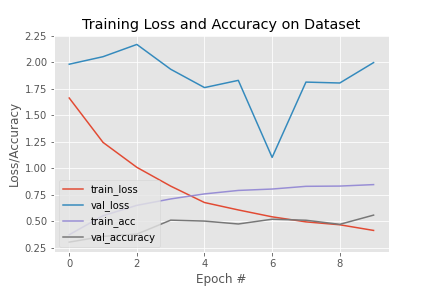
structure:

1. ResNet50
2. AveragePooling2D
3. Flatten
4. Relu
5. Relu
6. Dropout 0.5
7. Softmax

****

*Loss:1.4172 , Accuracy: 48%, Val loss: 1.4955, Val accuracy: 0.371*

*Learning rate: 0.001, Momentum: 0.9, Decay:0.01, Epochs: 10*

****

*Loss: 0.4141 , Accuracy: 84%, Val loss: 1.9960, Val accuracy: 0.5575*

*Learning rate: 0.05, Momentum: 0.9, Decay:0.01, Epochs: 10*

Our final choice for the algorithm structure is:

1. ResNet50
2. AveragePooling2D
3. Flatten
4. Relu
5. Relu
6. Dropout 0.5
7. Softmax



*Loss: 0.3, Accuracy: 87%, Val loss: 6.3, Val accuracy: 0.3*

*Learning rate: 0.01, Momentum: 0.9, Decay:0.001, Epochs: 10*

We concluded that this worked the best in our case.

With the same algorithm but 50 epochs, we reached an accuracy of 96%.

**5 Discussions**

We acknowledge that we can get even better and faster results when using object detection prior CNN. The main idea is to find the relevant object in the frame and learn on it, instead of the whole frame with lots of noise. Another great way would be using a motion/action detection algorithm with RNN, for example by creating a skeleton over the relevant object and learn its movement. Unfortunately, these solutions are time-consuming and require stronger hardware.

As further project development and improvement, we are planning to add a scoring feature. Initially, we planned to implement this algorithm on live stream (e.g. webcam recording) video, but it’s not possible due to the fact that we extract frames from the video which takes time and we risk losing lots of important data (skipping frames).

**6 Conclusions**

We showed that it is possible to classify video using quite a simple algorithm with CNN. It is possible to get great results with little time and limited data and home equipment. We also saw that an increasing number of train data and training period drastically affects classification accuracy.

**References**

1. K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385,2015.
2. He, K., Zhang, X., Ren , S., Sun, J.: Identity Mappings in Deep Residual Networks. In: ECCV (2016)
3. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEEConference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778
4. Video reference:
   1. Ladies Free Skating 2019 ISU - <https://youtu.be/FiSLEHITeNw>
   2. Men Free Skating 2019 ISU - <https://youtu.be/zJ42qYcKwbI>
5. Sagarvegad, Video Classification CNN and LSTM, <https://github.com/sagarvegad/Video-Classification-CNN-and-LSTM->
6. A. Rosebrock, Video classification with Keras and Deep Learning, 15 July 2019