Leabra

1st Matthew Schroder
Department of Computer Science
Middle State Tennessee University
Murfreesboro, TN USA
mrs8n@mtmail.mtsu.edu

2nd Justin Cao
Department of Computer Science
Middle State Tennessee University
Murfreesboro, TN USA
jmc2ei@mtmail.mtsu.edu

3rd Bradley Carter
Department of Computer Science
Middle State Tennessee University
Murfreesboro, TN USA
bec3x@mtmail.mtsu.edu

4th Sean Gately
Department of Computer Science
Middle State Tennessee University
Murfreesboro, TN USA
scg3q@mtmail.mtsu.edu

5th Christopher Sutton
Department of Computer Science
Middle State Tennessee University
Murfreesboro, TN USA
cs6t@mtmail.mtsu.edu

***Abstract*—The objective of our purposed neural networks is to develop a methodology to model and evaluate the accuracy and performance of a particular exercise, in this case the Squat. With our ultimate end game goal being able to live feed forward images using external hardware such as a webcam or a camera. To achieve this/these goal(s), our neural network will be using a convolutional network approach as this ideal with image data input and classification and prediction problems. At this point our input data set will be acquired thought the use of an image scrapper in conjunction with images from google images. After acquiring our images they will be reformatted to numpy arrays before being passed to our neural network. Key layers of our convolutional networks and hyper parameters to note are image sizes of 150 by 150, two conv2d layers of 64 and 128, dropout rates of .7 and .6. Batch sizes of 32 and epochs of 50. These layers and parameters allowed ran more consistently and kept the network and images simple enough to prevent too much overfitting. Results included categorical accuracy in the high nineties even as high as 98 percent with validation also in the high eighties. All while maintaining losses at low percent of single digit or low teens percentage.**

***Index Terms*—convolution, physical, exercise, motion, performance, form, technique, therapy**

I. Introduction

The use of neural networks and machine learning in conjunction with human motions is hardly a new application. Neural networks have been seen to go hand in hand with human motion applications such as playing key roles in physical activity for performance, efficiency and also in healing and rehabilitation such as physical therapy.

The objective of our research is to develop a neural network and a methodology for modeling and evaluating certain physical activity and exercising. For the purpose of our initial research we will be focusing on the squat exercise in particular.

The squat is a very popular common compound exercise with experienced and new gym goers alike. That focuses on many key muscles and that is often done with the use of additional weights, and with many variances in stance and form. Hence why when not done correctly or with proper technique an individual can be prone to numerous short and long term injuries. Thus we will be focusing on the squatters form and technique.

II. Background

The purposed neural network's ultimate goal is to take live images from external hardware such as a camera or webcam and feed forward that data into our network to tell how ”proper” the subject's squat technique and form is.

To achieve this goal our team choose to use a convolutional network. As our intended input was to be feed forward images inputs this was an easy choice. Convolutional Networks excel for image data inputs, and classification predictions problems. [1] They are also designed to be invariant to object position and distortion in the image scene which in dealing with form and technique in physical exercise is ideal. [2]

III. Methods

In this section we will go over our baseline methods how we navigate and train our neural network to achieve our desired end game goal. But it is important to note first and foremost we wanted to achieve a neural network that could at the very least determined whether our image input was a squat at all, I.e. a squat detection network.

A. Data Set

To begin we first needed to decide on a means of an initial data set of inputs. We chose to implement an image scrapper to go conjunction with Google images to achieve this. Using this method, we amassed roughly a total of 1200 images give or take of images of physical activities such as walking, running, benching etc. which were all considered ”non” squats at this point. Alongside roughly 300 actual squat images.

Key points to note, our pool of images was not exclusively grayscale or reformatted to be, this was mainly to not bottleneck or limit our available images in anyway as we were concerned about quantity of images available from our image scrapper and google. Our data set of images were however resized to a 150 by 150 pixel scale. Also, it is important to note that our data set of images truly needed to be shuffled or randomized. While this is a general rule with neural networks in general to prevent bias or predictable learning, it was particularly important or very apparent with our data set size. [3]. And our images lastly needed to be reshaped and formatted to a NumPy array before being passed to our network.

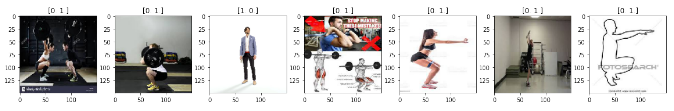
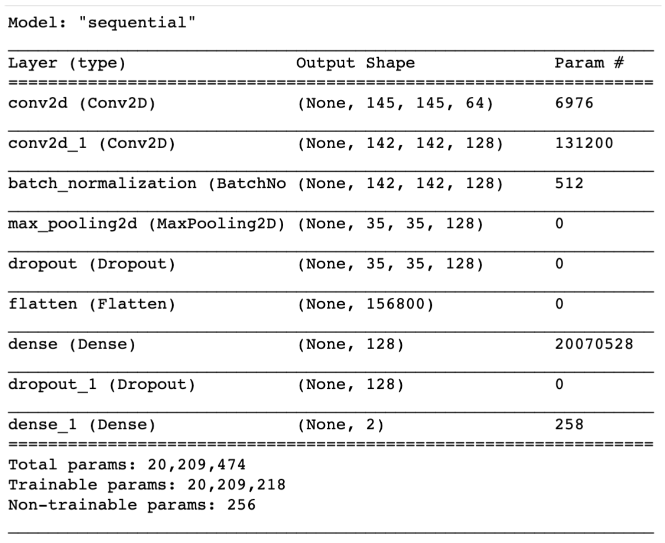


Fig. 1. Examples of our image data set.

B. Convolutional Network

Moving onwards to our choice of Convolutional Neural Network and it's architecture. To recapitulate on why we choose a convolutional network briefly, it is the go to for when dealing with image sets and classification problems. Convolutional Neural Networks preserve the spatial relationship of pixels of said image, features, and weights are learned and reused a cross all pixels and layers. This reusability is key. [4].



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 145, 145, 64)      6976
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 142, 142, 128)     131200
_________________________________________________________________
batch_normalization (BatchNo (None, 142, 142, 128)     512
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 35, 35, 128)       0
_________________________________________________________________
dropout (Dropout)            (None, 35, 35, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 156800)            0
_________________________________________________________________
dense (Dense)                (None, 128)               20070528
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 258
=================================================================
Total params: 20,209,474
Trainable params: 20,209,218
Non-trainable params: 256
_________________________________________________________________
```

Fig. 2. Model Summary

Now to begin breaking down our convolutional neural network's architecture. Following from the above figure. We are using two convolutional layers of size 64 and 128 with kernel sizes of 6 by 6 and 4 by 4 respectively. The purpose of these layers is to begin capturing high level features of our input such as edges and such [4].

We then implement a batch normalization layer. The purpose and benefit of this layer is to not only standardize our inputs but to also to accelerate the training of our network [5] [6].

Following along is our pooling layer. Also called subsampling or down sampling [7]. This layer works in hand with the previous cov2d layers to dimensional reduce spatial size of previous features and weights while retaining the most important features/weights [4]. Essentially a filtering layer.

The two dropout layers that are coming next serve are a very precisely simple means to help prevent our network from overfitting, and tuning the inductive bias. [6].

Lastly are our dense layers of our Convolutional Neural Networks. Within these layers and Keras are we it handles our output layer. The softmax activation makes our output sum to 1 so we can interpret it as possibilities [8]

IV. Results

In this section includes our most recent implementation and training of our network's output. Hyper parameters to note that we used include. Convolutions layers of 64 6,6 and 128 4,4. Momentum of .50 within the batch normalization layer. Dropout layers of .6 and .7. The following graphs show different outputs Accuracy and Loss with different Batch sizes and epochs.

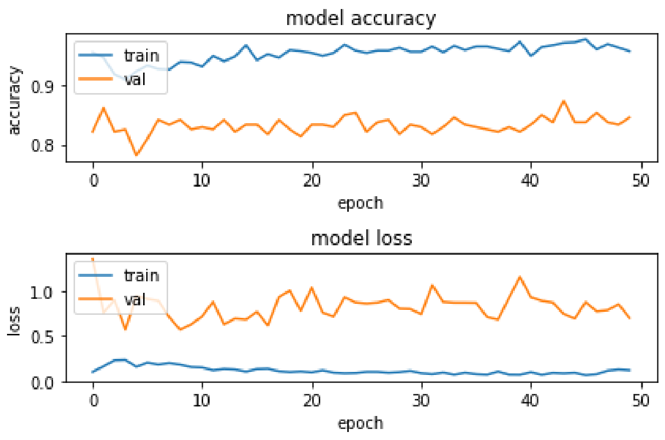


Fig. 3. Batch Size 32 and Epochs of 50.

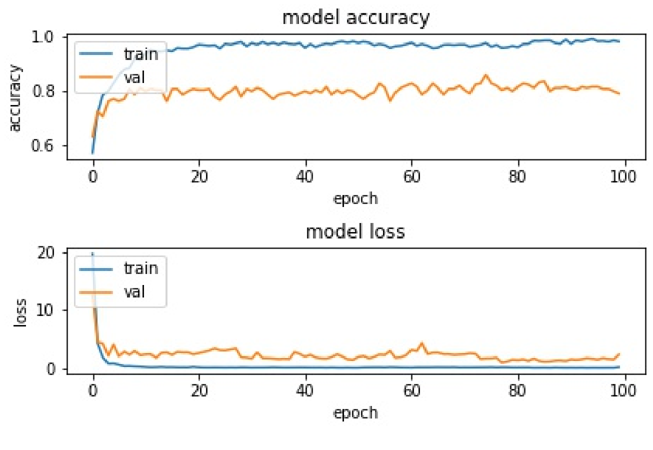


Fig. 4. Batch Size 50 and Epochs of 100.

V. Discussion

Here we are restating our goals and progress. To that we believe we achieved a neural network that at the very least

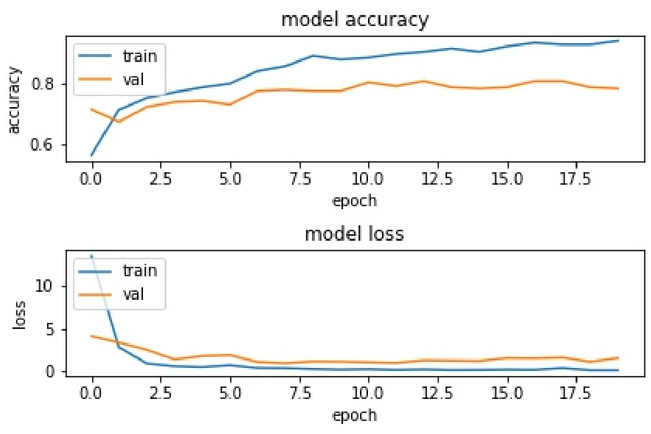


Fig. 5. Batch Size 50 and Epochs of 20.

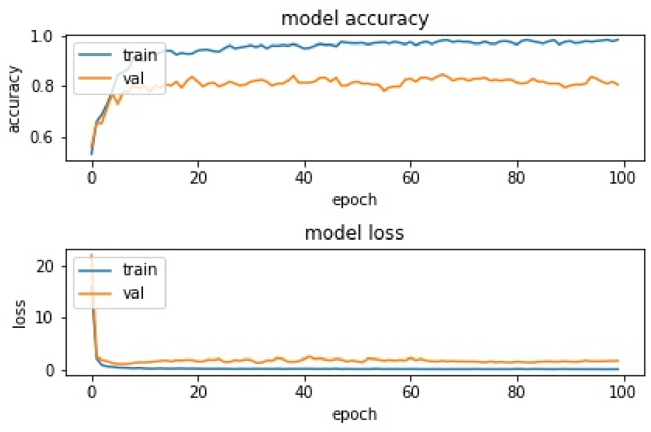


Fig. 6. Batch Size 100 and Epochs of 100.

is learned and trained to be a squat classification network. To progress our neural network further we will be looking to minimize our losses even further. Future steps would be to implement methods to measure the performance or correctness of said squat input.

REFERENCES

[1] J. Brownlee. When to use mlp, cnn, and rnn neural networks. [Online]. Available: https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/

[2] ——. Crash course in convolutional neural networks for machine learning. [Online]. Available: https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/

[3] L. Murgai. Reducing bias in ai. [Online]. Available: https://blog.insightdatascience.com/reducing-bias-in-ai-d64bc3142ae2/

[4] S. Saha. A comprehensive guide to convolutional neural networks... [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53/

[5] S. Saurav. Batch normalization in deep neural networks. [Online]. Available: https://www.kdnuggets.com/2020/08/batch-normalization-deep-neural-networks.html/

[6] C. Garbin. Dropout vs. batch normalization: an empirical study of their impact to deep learning. [Online]. Available: https://link.springer.com/article/10.1007/s11042-019-08453-9

[7] Ujjwalkarn. An intuitive explanation of convolutional neural networks. [Online]. Available: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

[8] E. Allibhai. Building a convolutional neural network (cnn) in keras. [Online]. Available: https://towardsdatascience.com/building-a-convolutional-neural-network-cnn-in-keras-329fbbadc5f5