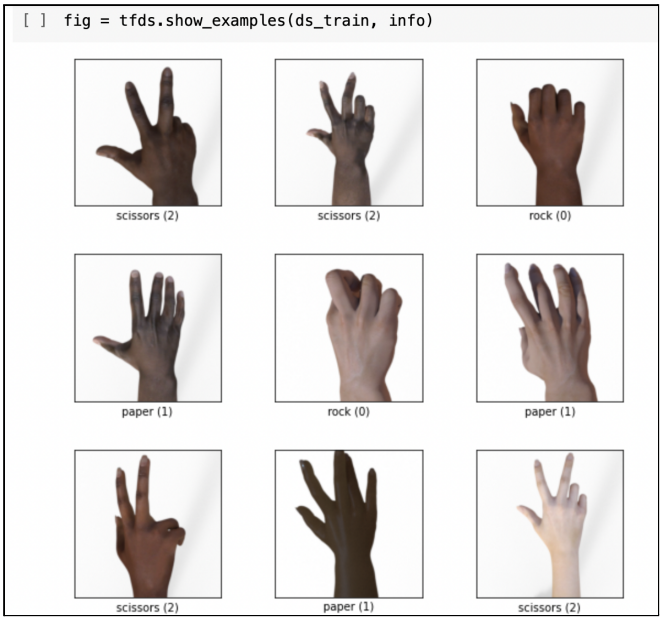
Image Classification with a Convolutional Neural Network

Amy Hardy A16558949
COGS 118B Fall 2021

I. Introduction

Deep Learning, a subcategory of Artificial Intelligence and Machine Learning, has made significant advances in the last few decades and is critical for technologies such as facial recognition, automated driving, natural language processing, computer vision and more. It relies on the usage of neural networks, which are biologically inspired models that are capable of taking in extremely large amounts of input data to make accurate predictions. These applications of deep learning can be used to vastly improve public safety, healthcare, financial services, retail businesses, and customer services. Artificial Neural Networks (ANN) consist of multiple layers, creating levels of abstraction within the data to efficiently identify objects and patterns. One of the most powerful types of deep neural networks for image classification is the Convolutional Neural Network (CNN), which are especially useful for detecting patterns and making sense of them. The difference between a CNN and a traditional ANN lies within it's added convolution layers. These layers receive input and transform it in a way that the next layer will be able to interpret. These layers contain a certain number of filters, which are individual (n x n shaped) matrices that convole over smaller grids of pixels within the image. The dot product of the filter matrix and grid matrix is calculated and the result is a more condensed version of the original image, with unique features in mind for each filter and is passed to the next layer as input. Tensorflow, a library for machine learning and artificial intelligence, can be used to efficiently produce traditional and convolutional neural networks. The dataset I used to explore image classification is the "Rock, Paper, Scissors" dataset made available by Tensorflow. It consists of 2892 CGI-based images, pre-split into training and testing data. They were created with a diversity of skin tone, hand size, nail polish, and poses. As seen in Figure 1, most of the images are very easy for humans to recognize, although some are slightly ambiguous. Because this is a



Tensorflow dataset, it is already moderately optimized for this type of project and took very little data cleaning to get started. To prepare the data, it first had to be categorized into images and labels. The labels were rock (0), paper (1), and scissors (2). These variables then had to be converted from Tensorflow tensors to NumPy arrays. The train image arrays had a shape of (2520, 300, 300, 3) for 2520 total training images, each with a size of 300 x 300 and 3 values of color. This could be reshaped into (2520, 300, 300, 1) because the problem at hand focuses on edge detection and we can omit the color channels

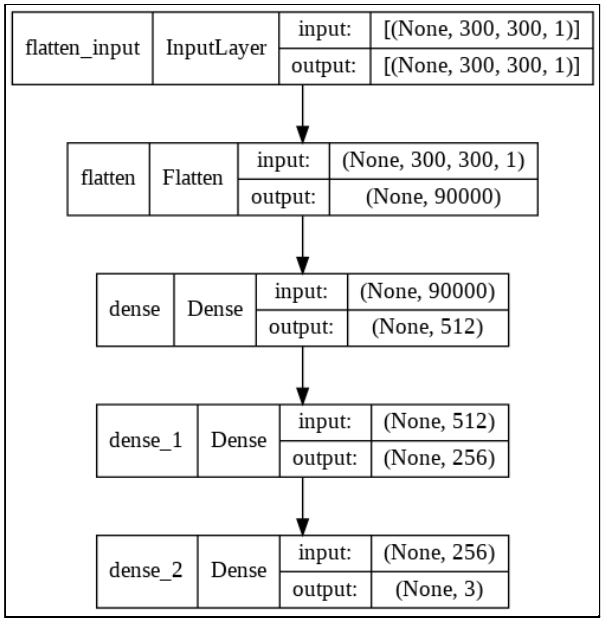
and therefore reduce the amount of input to the network. Lastly, I normalized the pixels of the training images and testing images by dividing by 255 so the RGB value would be recognized by the network as a number between 0 and 1.

II. Related Work

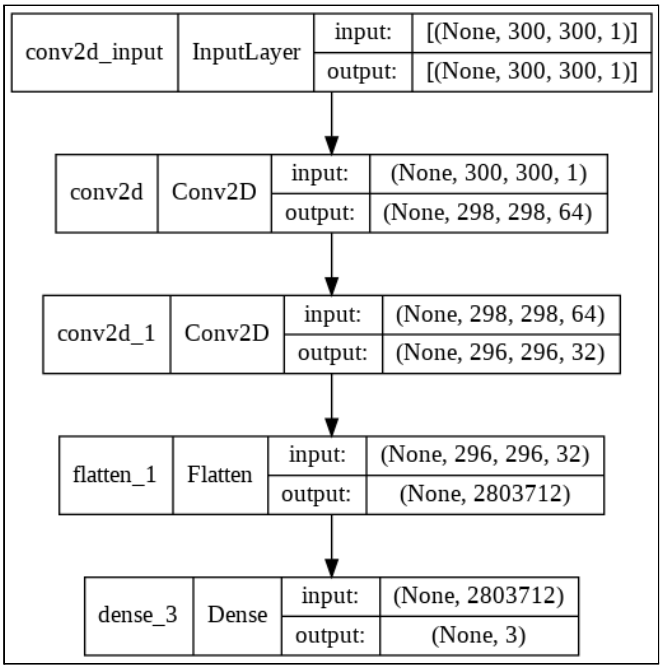
As stated, image classification and advances in these types of convolutional neural networks can be used in the medical field to help understand diseases, their causes, and how to identify them quickly and efficiently. In 2014, Quig et al. developed a convolutional neural network algorithm that could "automatically and efficiently learn the intrinsic image features from lung image patches that are most suitable for the classification purpose" (Quig et al. 2014). They stated that their model could be generalized to categorize and identify other diseases as well. Improving these algorithms and finding new ways to use them can result in massive steps forward when it comes to diagnosing and treating diseases.

III. Methods

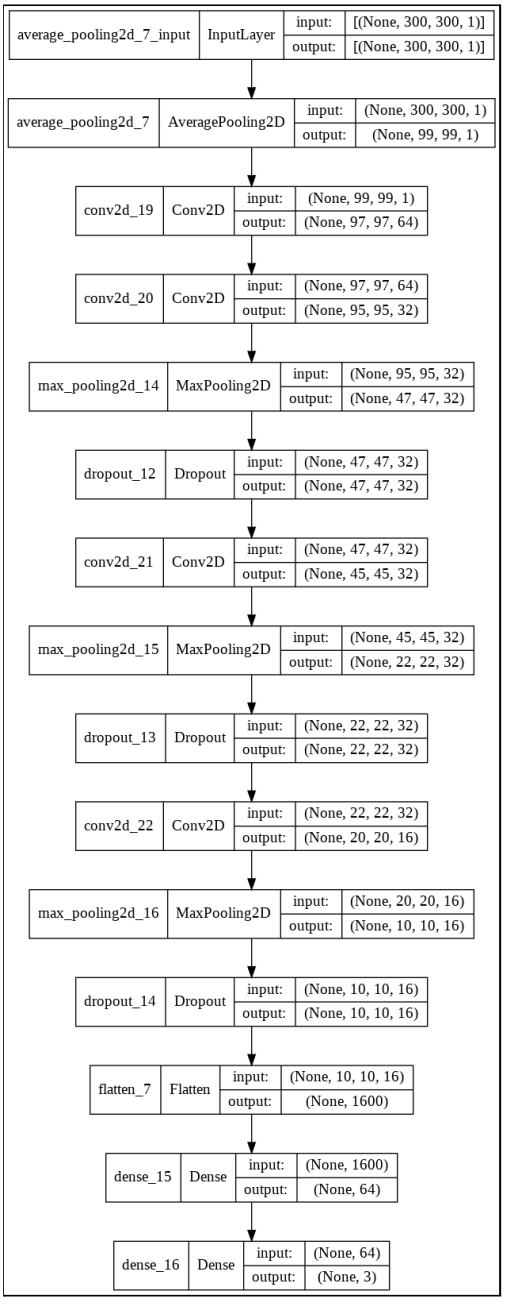
To better understand the differences between traditional artificial neural networks and convolutional neural networks, I decided to first start with an ANN and examine how it performed and where it could do better. To do this, I used a Keras Sequential model with four layers. The first layer was a "flatten" layer which transformed the 300 x 300 image into a single column of dimension 90,000. The next three layers were fully connected layers with 512, 256, and 3 filters, respectively. The second and third layer used rectified linear unit activation while the fourth layer used softmax activation. To compile the model, I defined the appropriate optimization and loss functions and continued to fit the model.



Next, to create a convolutional neural network, I took the code for the traditional neural network and swapped out two of the fully connected layers for 2 dimensional convolutional layers, with 64 and 32 filters. Finally, for the last version of the model I decided to increase the amount of total layers, adding an average pooling layer, three max pooling layers, and three dropout layers.



These act to reduce the spatial size of the input image and inturn reduce the complexity of the network. The dropout layers regularize the model and help to reduce overfitting. After I completed these models, I used hyperparameter tuning to see if I could improve the final model in any ways. The tuner does this by iterating through different options for number of layers, number of filters, etc. The final change I made was adding layer weight regularizers (L1 and L2) to the last layer.



IV. Results

For the first instance of the model, the traditional ANN, we got a training accuracy of 79% and a testing accuracy of 50%. This result reveals that there is obvious overfitting happening within the model. This not improved with the first convolutional neural network, where we got a training accuracy of 100% and a testing accuracy of 55%, an even larger gap between them. After manually testing different combinations of layers and hyperparameters, I ended with a final result in the improved CNN of training accuracy 94% and testing accuracy 82%. Unfortunately, the Keras tuner was not able to give me any results higher than that, so I decided to stick with my final model and disregard the tuning.

IV. Discussion

The most important lesson learned from this project is that although a model may perform well on training data, that does not mean it will perform well on the testing data, and that is where it needs to be accurate. This is known as overfitting, and when it occurs at too large of a degree, the model is not useful. Seeing the differences between the first CNN and the improved CNN showed that I was able to decrease the amount of overfitting, and although it was not perfect, resulted in a model that could correctly predict the label of the image most of the time. I decided to add the layer weight regularizer to the last layer of the final model because according to “Pattern Recognition and Machine Learning” adding regularization to neural networks can decrease overfitting. I believe this step and adding additional pooling and dropout layers was what ended up decreasing the amount of overfitting (Bishop 257). I plan to continue working on this model, with a goal of further reducing overfitting by exploring more ways to implement data augmentation, regularization, and reducing input complexity.

VI. References

Bishop, Christopher M. Pattern Recognition and Machine Learning. New York :Springer, 2006.

Galli, K. (2020) neural-nets/real_world_example source code [Source code]. https://github.com/KeithGalli/neural-nets

Howard, A. G. (2013). Some improvements on deep convolutional neural network based image classification. arXiv preprint arXiv:1312.5402.

S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), 2017, pp. 1-6, doi: 10.1109/ICEngTechnol.2017.8308186.

T. Guo, J. Dong, H. Li and Y. Gao, "Simple convolutional neural network on image classification," 2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA), 2017, pp. 721-724, doi: 10.1109/ICBDA.2017.8078730.

Q. Li, W. Cai, X. Wang, Y. Zhou, D. D. Feng and M. Chen, "Medical image classification with convolutional neural network," 2014 13th International Conference on Control Automation Robotics & Vision (ICARCV), 2014, pp. 844-848, doi: 10.1109/ICARCV.2014.7064414.