Christian Shadis

Dr. Alsallami

CS-383

Due 5/9/2021

Detecting Cancer with a Convolutional Neural Network

Melanoma, nevus, and seborrheic keratosis are three different types of skin lesions. Melanoma is a lethal skin cancer, while the other two are benign. For my final project, I decided to use a convolutional neural network to train a Keras model to classify these three skin lesions. I used Python, Keras, Jupyter Notebooks, and Pyspark on top of an AWS Elastic MapReduce cluster running Spark to build the final version of this model.

When I started this project, I had never worked in-depth with machine learning, neural networks, or Keras before, so the first step to completing the model was to learn about convolutional neural networks and how to build them in R. This lack of exposure exponentially increased the amount of time I had to spend learning the tools and technologies needed. I planned on building the model in R, creating an AWS Elastic MapReduce cluster, installing RStudio Server on the master node, and running the model that way. I gained an introductory understanding of how convolutional neural networks work, and I successfully built the model in R. Migrating this R script to the cloud, however, presented a challenge that was too large for me to overcome. After many hours, I made the decision to translate my model into Python, since Python's integration with AWS, especially Spark, seemed to be better than R's.

I was able to translate the model into Python without much difficulty, but once again hit roadblocks trying to migrate the script to the cloud. I tried bootstrapping the AWS cluster with a Bash script to install Python dependencies on all nodes, but when I submitted the Pyspark job those dependencies would be missing. After a staggering amount of trial and error, I gave up on this venture

as well. I tried to install dependencies with both Pip and Anaconda with no luck, and I tried packaging dependencies as a Conda archive package and exporting the package into the spark job with no luck. Finally, I combined a Python script with a Jupyter Notebook attached to an Elastic MapReduce cluster that would read in images from S3, and installed python dependencies using Spark's 'install_pypi_package' method. This combination of tools was successful.

The data for this model comes from a Kaggle dataset containing over 2,000 high-definition .jpg images of the three skin lesions, organized by which lesion it is. The data had to be processed in order to be passed into the model, and I wanted to use a Keras ImageDataGenerator to augment images and increase the size of the training set. The first step in this process was to write a Python script to resize all the images to 100x100, then pass those images through the ImageDataGenerator. I then uploaded all these picture files to S3 for use in the cloud-based model. In order to analyze the pictures, I used Pillow and Numpy to convert each image into an array of pixels before creating the Keras model and training it using those arrays. When the model is finished fitting, an output file is appended with the sample size, accuracy, and exact date and time of the model's execution, and uploaded to S3.

Using a training set of 1000 images for each melanoma, nevus, and seborrheic keratosis, running the validation set through the model yielded a prediction accuracy of 48%. This is higher than I would expect for my first model since a pure guess would have a prediction accuracy of about 33%. However, I plan on improving the accuracy of the model by running it with higher-definition images, and more augmented images in the training set. I will also tweak the values of various hyperparameters to maximize accuracy. Now that the model is running on an EMR cluster instead of my local machine, performance limitations are mitigated, and I have more freedom to run compute-intense operations.

If I could do this project over, I would have found more appropriate tools for the job. Trying to run an image recognition model using Spark, which is meant more for large calculations over many rows

of data, was not intuitive or easy. I would find a tool that is meant for image processing and use that. Choosing the wrong tools for the job cost me a lot of time and effort – I ran 118 different EMR clusters before I was successful. I would also dedicate some of the project time to parallelizing processes. My original R script model had a very expensive for-loop that could have been parallelized using the doParallel package and the foreach operator, but translating that into Python removed the parallelization. The most expensive operation in the final model is the training of the model, which cannot be easily parallelized. Perhaps by using different tools I would have been working in an environment that supports parallelization more easily.

The most valuable information I learned was about convolutional neural networks. My understanding is still relatively elementary, but I am extremely interested in machine learning and I was finally able to build, implement, and understand the convolutional neural network. Being introduced to Keras, a very large machine learning library, also gives me a head start on the next machine learning project I decide to attempt. Not only did I learn from this project, but it has inspired me to undertake further image recognition projects.

Projects such as this one have the potential of being extremely valuable. Successfully training a computer to identify similar-looking lesions as melanoma would be incredibly helpful to oncological diagnoses. Because of the real-world use case, I plan on developing this project further and improving the accuracy drastically – this should be easier now that the model is up and running on AWS. Another reason to stick with this project beyond the scope of this class is my attraction to Machine Learning, the subject in computer science most appealing to me. Regardless of the future accuracy of my model, the project was a success in that I learned a new concept from scratch in two different languages, and successfully implemented a cloud-based model that will now be capable of higher accuracy when trained on a larger number of higher-resolution images.

Research Materials:

Allibhai, E. (2018, November 15). Building a convolutional neural NETWORK (cnn) in Keras. Retrieved May 08, 2021, from https://towardsdatascience.com/building-a-convolutional-neural-network-cnn-in-keras-329fbbadc5f5

Convolutional neural networks in Python. (n.d.). Retrieved May 08, 2021, from https://www.datacamp.com/community/tutorials/convolutional-neural-networks-python

Convolutional neural Networks: A Python tutorial using TensorFlow and Keras. (n.d.). Retrieved May 08, 2021, from https://www.kdnuggets.com/2019/07/convolutional-neural-networks-python-tutorial-tensorflow-keras.html

Fernandez, A. (2021, February 14). How to build a convolutional neural network in Keras. Retrieved May 08, 2021, from https://anderfernandez.com/en/blog/how-to-create-convolutional-neural-network-keras/

Rohrer, B. (Writer). (2016, August 18). *How convolutional neural networks work* [Video file]. Retrieved April 19, 2021, from https://www.youtube.com/watch?v=FmpDIaiMIeA

Documentation:

Pyspark: https://spark.apache.org/docs/latest/api/python/

AWS: https://docs.aws.amazon.com/emr/index.html

Pillow: https://pillow.readthedocs.io/en/stable/

Boto3: boto3.amazonaws.com

Repository available at github.com/ctshadis/cs383-finalproject