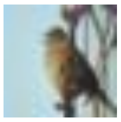
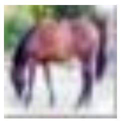
Jason Boyd
A02258798
December 18, 2020

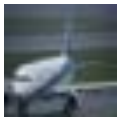
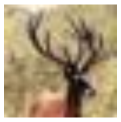
Training and Analyzing Intelligent Systems on CIFAR-100

The purpose of this project was to design, train, and analyze three best performing neural network types on the CIFAR-100 image classification dataset. Those network types included building convolutional, artificial, and random forests using the tools available with TensorFlow, TFLearn, and SKLearn. This report shows the findings, successes and failures, the program process, and reflections from building the project and training those networks. For more information and details on the project itself, please refer to the README.md file included with the project supplied with this report or visit the hosted repository https://github.com/itsjaboyd/cifar100-project through GitHub.

CIFAR-100 Dataset

The CIFAR-10 dataset can be found and downloaded at https://www.cs.toronto.edu/~kriz/cifar.html. Quickly overviewing the dataset, CIFAR-100 consists of 100 classes containing 600 images each (500 training images and 100 testing images) for a total of 60,000 images. Those 100 classes are divided into 20 super-classes. Each image is labeled with its immediate and super classes. CIFAR-100 exists as a subset of the 80 Million Tiny Images dataset that serves as a major dataset to perform object image classification techniques on. The dataset has somewhat of a twin known as CIFAR-10 in a much tighter knit and closer identification realm, including just 10 classes with 6,000 images each instead [3]. Some example images are given by the dataset below.

CIFAR-100 has been a rather popular dataset to work with in devising highly accurate convolutional neural networks. For example, the latest and greatest model for this dataset comes from Alexander Kolesnikov et al. using a newly proposed method known as Big Transfer (BiT) and other heuristics for achieving their highly accurate network [1]. The second highest accuracy has been achieved by Mingxing Tan and Quoc V. Le proposing a new scaling method and highly effective compound coefficient on already existing networks [2]. There exist many other impressive and notable network architectures on the CIFAR-100 dataset that are interesting to review, but of course being extremely popular in the image classification community, there are a plethora of classification methods out there.

After researching and experimenting with the official downloadable dataset, it turns out that TensorFlow has prepackaged built-in datasets for users to explore, and CIFAR-100 happens to be one of them. Instead of manually downloading the dataset, the project makes use of the TensorFlow version, which makes the source code cleaner and easier to maintain. After the first run of the executive driver within the project structure, TensorFlow automatically downloads the dataset for the user and sets it up within the project.

Convolutional Neural Network

Messing around with convolutional architectures is costly because they take significantly longer to train, however there was a good starting to point considering there were great examples in the lecture notes and with project one. The convolutional section of the project contains creating and loading functions for the best performing architecture and then deeper and shallower architectures. Arriving to the best performing architecture was a repeated test and fine-tuning process, closely watching the accuracy and validation loss of the network.

Jason Boyd
A02258798
December 18, 2020

The architecture of the network begins with connecting the first convolutional layer with the input layer that includes a filter size of three and 32 total filters. All convolutional and fully connected layers in the convolutional and artificial networks use rectified linear unit (RELU) activation. The first convolutional layer is tied to its pooling layer which is then connected to another convolutional layer, this time with 64 total filters and another pooling layer attached. Then we arrive to the last and final convolutional layer with 128 total filters and its max pooling layer. Tied to the last convolutional layer is three fully connected layers, first with 256 nodes, second with 128 nodes, and as the output 100 nodes. The network uses stochastic gradient descent as its optimizer and categorical cross entropy for its loss and a learning rate of 0.01.

After attempting to mess around with the python package PyPlot and doing some research to get accuracy and loss graphs, it turns out that TensorFlow comes with a powerful feature known as Tensorboard. This feature allows users to actively watch how well their networks are performing while they are training which was super useful in seeing the performance of each network. The graphs and data analysis for these networks are gathered from Tensorboard. Unfortunately when the Tensorboard flag is turned on, training slows down by a factor of 10 to 15, which was hugely detrimental to giving time for adequate training. To keep training times consistent across the project, each network trained with Tensorboard on in order to get the statistics and numbers on the networks trained.

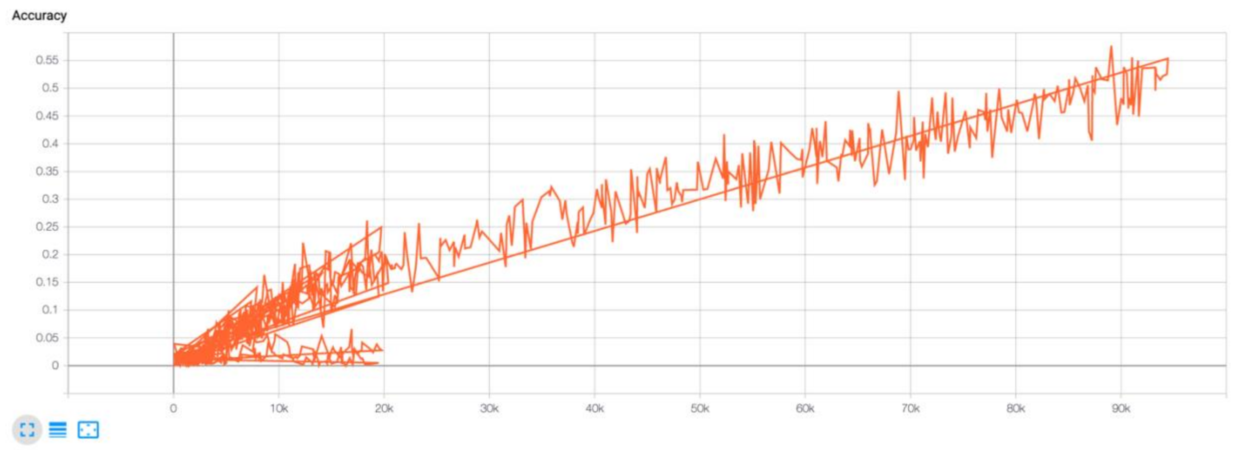


Figure 1: best performing convolutional net training accuracy.

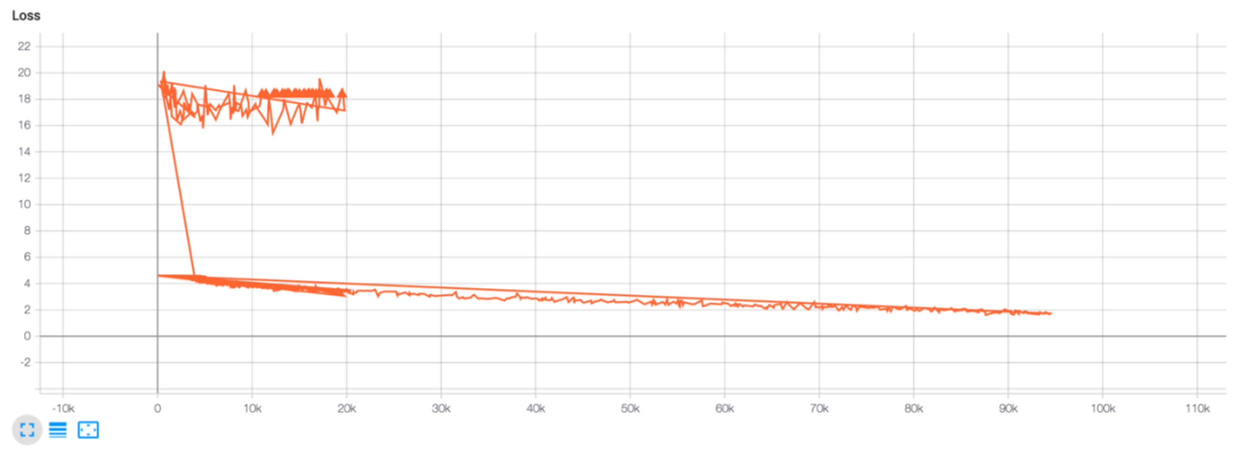


Figure 2: best performing convolutional net loss.

In terms of accuracy and loss, there was a very steady increase in accuracy and very steady decrease in loss, which is exactly what was needed to train a relatively okay performing network. The total training time before accuracy began to stop improving was a little over an hour and four minutes, the precise time being 04:19:31.953. The network was able to achieve roughly 37% accuracy on the testing data and around 36% accuracy on validation. Overall, this was the best performing network out of the rest of the network types.

Artificial Neural Network

Having a the fully connected layer capability, artificial neural networks are easily created and trainable within TensorFlow as well. The same story comes with artificial networks as with convolutional. The artificial section of the project comes with the best performing architecture as well as two other testing architectures, including a larger hidden layer network and a smaller hidden layer network. Again, arriving to the best performing architecture was a repeated process of testing and fine-tuning, closely watching the accuracy and validation loss of the network.

The architecture of the artificial network begins by connecting the input layer into the first fully connected layer of 512 nodes. Recall that with CIFAR-100 the images are 32 by 32 pixels, meaning that there are a total of 1024 pixels (inputs) and three-color channels being red, green, and blue (RGB), so the input layer size is effectively 1024 nodes. The second fully connected layer comes in with 256 nodes and is connected to the third and final output layer of 100 nodes. The network uses stochastic gradient descent as its optimizer and categorical cross entropy as its loss with a learning rate of 0.01. Again, here activations on hidden layer nodes are RELU and the outputs are softmax.

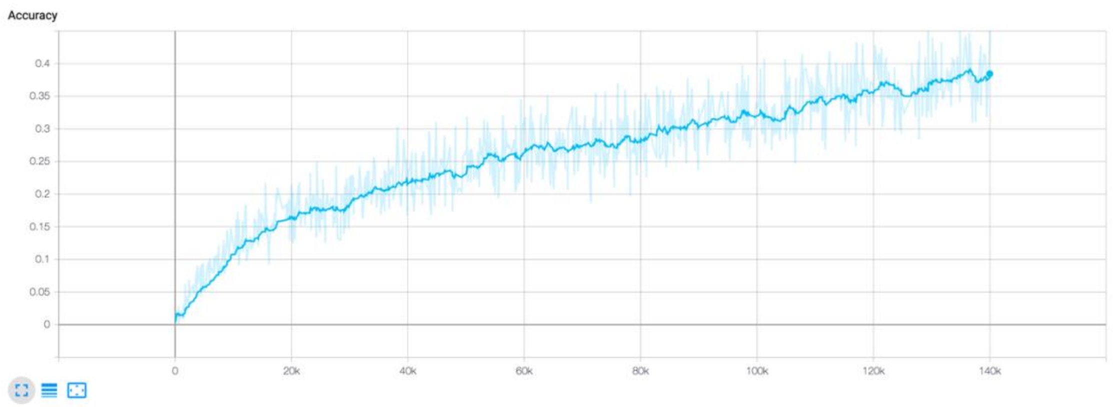


Figure 3: best performing artificial net accuracy

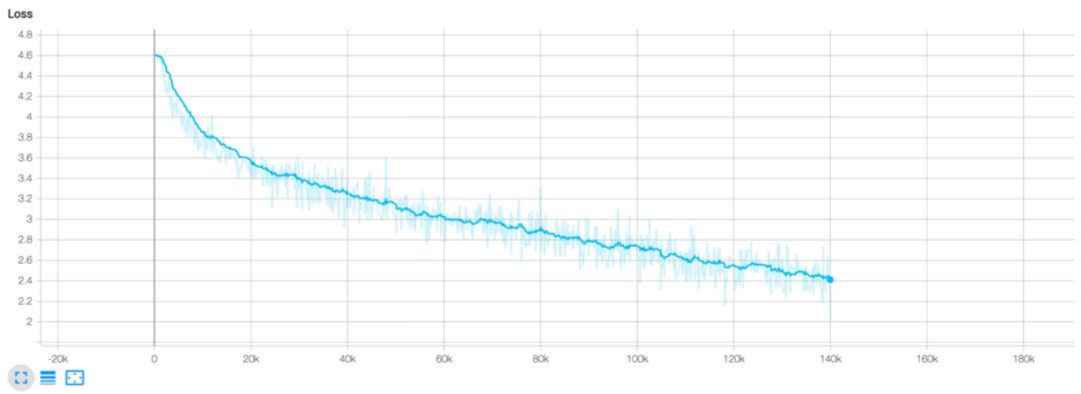


Figure 4: best performing artificial net loss

Jason Boyd
A02258798
December 18, 2020

In terms of accuracy and loss, accuracy improves quite quickly over a relatively short amount of training time and loss goes down just as quickly. The accuracy tops off at around 26% accuracy while the loss bottoms out at around 2.5. The calculated accuracies for the best performing artificial network include 26.11% validation and 26.98% testing. The total training time for this network came out to be just under three hours with an official time stamp of 02:52:23.717.

Random Forest Network

Unfortunately, due to the time constraints and effort the rest of the project has taken in development and training each network thus far, implementing the random forest section of this project could not be reached. While the required data for analysis will not be presented in the project, there will be estimations whereas random forests might compare to the trained convolutional and artificial neural networks on CIFAR-100. Beyond the scope of grading and submission of this project, development will most likely continue as results of the random forests would provide valuable insight into each of the image classification network types.

If random forests were to hold comparison with the other convolutional and artificial networks, they would most likely be a competitive contender for accuracy and training time. Referring to the in-class and homework assignment experience with random forests on the MNIST dataset, they trained incredibly quickly and possessed an extremely high accuracy, almost as close as the convolutional networks were able to achieve. An effectively tuned random forest would probably get around 30% or close to the accuracy of the artificial neural network. The training time with Tensorboard included would most likely take less time than both the artificial and convolutional networks, probably around the two-hour mark.

Project Reflection

More than a couple things were learned throughout this entire project experience. First and foremost, attempting to accurately estimate the time needed to build the foundational features of the project should have been given priority. Second, sometimes TensorFlow was rather unhelpful in outputting error messages and helping debug buggy code. Finally, rigorously sticking to the project schedule should have been one of the top priorities as well, given the circumstances and breaks in school.

Before beginning development on the project and as outlined in the project proposal, it seemed as though the structure and flow of the project was going to be simple. Three files for three different network types were going to be enough to organize the project, but after realizing that some code could be reused by other networks and unit testing could be helpful and how the project was going to be run exactly, re-structuring the project was necessary. A reiteration of the requirements phase was necessary to effectively plan out what was needed to complete, which minorly put back the project schedule. Allocating more than enough estimated time to gather requirements and seriously plan out exactly what the project should and should not perform is vitally important. It was extremely easy to get sidetracked and caught up in making cool but irrelevant features instead of focusing on the main task ahead.

While TensorFlow has decent documentation about using the framework, there were walls to push through and external research to be done. The biggest friend of a developer who is training and loading multiple different networks is the “reset_default_graph()” function. After what felt like days of trouble shooting checkpoint errors and fine-tuning network architectures, finally coming upon and using this function was huge in getting development back on track. At one point it was thought that completely retraining the saved convolutional network was necessary, but it in fact was not, which wasted time. Along with knowing the tools being used, doing adequate research into what other tools that could be used would have been helpful as well. TensorFlow ships with an API known as Keras for deep learning purposes that definitely could have come in handy in network analysis and training.

Every software developer should plan for setbacks within the project schedule, but between planning out milestones and deliverables, sticking to the schedule should be the number one goal. Over the project timeline was Thanksgiving break which was approximately a week off from school. Knowing myself and the nature of taking time of from school, absolutely no work was done over this Thanksgiving period, which of course pushed back the project timeline by another week. Luckily, there was around another two weeks budgeted towards the end of the timeline to catch up and work on the required documentation, but that week wasted was valuable development time.

While not every aspect of the project was completed, the project was extremely interesting and rewarding seeing these different types of networks perform in image classification. Even though the due date has come, project development will most likely not come to an end. Seeing the results of training a random forest on CIFAR-100 and comparing its results to the convolutional and artificial networks will be insightful and rewarding.

Conclusion

Convolutional neural networks remain the top competitor for image classification. While each type of neural network has their own advantages and disadvantages, the overall improvement in accuracy makes convolutional networks king in the image classification world. Random forests train incredibly quickly with a very high pay off in accuracy which make them a good quick solution for classification. Artificial neural networks are very simplistic in their nature as fully connected neurons that give decent accuracy in classification. Convolutional networks typically have the highest accuracy but take tremendously more time and power to train. This is seen with every popular image classification dataset out there; the most high-performance network types are consistently convolutional neural networks, and this is very much the case with CIFAR-100.

Jason Boyd
A02258798
December 18, 2020

Works Cited

[1] Kolesnikov, Alexander et al. “Big Transfer (BiT): General Visual Representation Learning.” ECCV (2020). https://arxiv.org/abs/1912.11370v3.

[2] Tan, M. and Quoc V. Le. “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks.” ArXiv abs/1905.11946 (2019): n. pag.

[3] Krizhevsky, A. (n.d.). CIFAR-10 and CIFAR-100 Datasets. Retrieved from https://www.cs.toronto.edu/~kriz/cifar.html