Tiny Imagenet - Applied Machine Learning

Michael Golfi
McGill ID: 260552298
michael.golfi@mail.mcgill.ca

Sorin Muchi
McGill ID: 260575810
sorin.muchi@mail.mcgill.ca

Srinivas Nadella
McGill ID: 260531213
srinivas.nadella@mail.mcgill.ca

Team name: "Highly Classified"

***Abstract*—This project studies the effectiveness of training different machine learning algorithms on a subset of 40 classes from the ImageNet dataset. Of the 40 classes, there are images of birds, dogs, cucumbers, pumpkins, at least two types of squash and other miscellaneous subjects. This project uses Logistic Regression, Feed-forward Neural Network with back propagation and finally a retrained convolutional Neural Network based on the Inception image set. Furthermore, we explore several techniques such as manipulating hyper-parameters, enriching the learning dataset, and the use of random image perturbations, in order to understand, their effects on prediction accuracy.**

I. Introduction

Classification is an important aspect of machine learning. Image classification can now be found everywhere in our digital lives. Some of the techniques illustrated in this paper, such as convolutional neural nets are used for satellite image analysis, facial recognition, cancer detection, image search and more. This experiment presents image classification on 40 categories of images originating from the ImageNet dataset. We use baseline learner, logistic regression and compare it against a feed-forward neural network and convolutional neural networks. We take this a step further exploring a retrained neural network.

II. Related Work

Some related experiments are the ImageNet [1] [2], and CIFAR-10 [4] competitions. Several industry leaders such as Google, IBM and Microsoft have made great advances in image recognition and detection through the use of convolutional neural network models [6].

III. Problem Representation

The problem can be represented in the following subcategories.

A. Pre-processing

The dataset provided was in the form of three numpy arrays. The training data was represented first by the instances of images, then the three color dimensions then each dimension of pixels. That is to say that the dataset had shape $(26344, 3, 64, 64)$. The training labels were stored in a 1-D array with the label numbered from 0 to 39. The test data was the same form as the training data but with only 6600 images.

For parts 1 and 2, the numpy arrays were used as raw inputs. In part 3, the training and test data were converted back to image format for the purposes of training, validation, and testing. This was also done for the purpose of visualizing the dataset during our initial investigation. It gave a sense of how the images appeared, along with their pixel density and the challenges needing to be overcome for the experiment.

B. Perturbations

A common method for both extending an existing data set and boosting accuracy on unseen test data, is to create multiple images within a training set using perturbations. Perturbations can include transformations such as mirroring, cropping, scaling, flipping. The convolutional neural net used in this paper used perturbations including flipping in all four directions, brightness, resizing nearest neighbour, hue, and image standardization. The results of this are discussed in Section VI: Testing and Validation.

C. Image Recognition

Image recognition is a classification problem which groups unlabelled images into predefined classes. This is an ever-increasingly important task due to the growing popularity of inference of meaning for image data by companies such as Facebook and Google, to name two. Image data is hard to classify due to the nature of the raw pixel data representing the images. Image metadata also increases the difficulty of the problem since size, colors or grayscale, and pixel density all will affect the nature of techniques used to label the data.

The image classification techniques used in this paper, and their associated challeges, namely logistic regression, feed-forward neural networks, and convolutional neural networks, have been widely addressed in the literature.

IV. Feature Selection and Extraction

In this experiment, pixel values were used as input features with each pixel representing a feature as a 3-tuple of red, green and blue values (RGB). The baseline regression, feed-forward network and retrained network all used a one dimensional representation of the data supplied in the numpy data provided where each pixel was sequentially represented and each pixel value was flattened in RGB order.

$$D = \{(p_{1r}, p_{1g}, p_{1b}), (p_{2r}, p_{2g}, p_{2b}), \ldots, (p_{Nr}, p_{Ng}, p_{Nb})\}$$

V. Algorithm Selection and Implementation

Algorithm selection was not necessary in the first two parts of the experiment as these algorithms were predetermined. In the third part of the experiment, retraining an existing neural network was deemed necessary to save on overall execution

time and to acquire performance gains from existing neural architecture.

The Inception V3 model was readily available as a retraining model from the Tensorflow documentation [11]. The architecture of the model can be visualized in figure 9.

An important aspect of the neural network used is the role of convolutions as an integral operation in image processing. Convolutions in image processing will intensify dissimilar boundaries. Figures 1 and 2 show the effects of convoluting an everyday image. The convoluted image has dissimilar boundaries highlighted the most. For instance, the boundary of the skin and the shirt is a very abrupt color change, as are the window frames in the background. These lines in the convoluted image happen to be the brightest lines. Likewise, the most similar regions are darkened. The entire area of the black shirt is also black in the convoluted image. This is an important concept to consider since this will highlight the outlines of shapes in images, and ultimately be what guides any recognition. Several layers in our retrained model perform convolutions to increasingly remove noise from the image as our goal is to just look at shapes. Each subsequent layer in our network gains more knowledge about the shapes in the image data.

Fig. 1. Original Image

A. Logistic Regression

Logistic regression was implemented using Scikit-learn libraries. This algorithm was not chosen since performance was poor despite choosing different pairings of hyper-parameters. Overall, logistic regression yielded performance of approximately 20% on the validation set.

Fig. 2. Convolution applied to Image

B. Feed Forward Neural Network

Feed forward neural networks are the simplest form of artificial neural network architecture. In this architecture, information moves only in one direction (forward) through layers of perceptrons. For this experiment, each picture was flattened into a 1 x 12,288 dimensional array. Our feed-forward neural network was kept simple in the interest of time. It was composed of a first layer of 12,288 nodes, a hidden layer with 12,288 nodes and finally an output layer of 40 nodes. [12]

C. Gradient Descent

For gradient descent algorithm, stochastic gradient descent was used with minibatches. Gradient descent traditionally computes the gradient using the whole dataset. Instead of this method, stochastic gradient descent (SGD) computes the gradient using a sample. There are many benefits to this method, mainly in computation time. Besides computation time, SGD performs much better for error manifolds that have lots of local maxima and minima. Minibatches were introduced to avoid the noise problems of SGD. Using simple samples to calculate the gradient results have a tendency to be noisy. Minibatches take a sample, allowing the smoothing of noise in the calculated error. The formula for stochastic gradient descent is as follow:

$$w_k \to w'_k = w_k - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial w_k}$$

$$w_k \to b'_k = b_k - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial w_b}$$

Where w is the weights, b is the bias η is the step size, m is the size of the mini-batch, and C is the cost function.

Stochastic gradient descent works by picking a random mini-batch of inputs, and training with those, where the sums are over all the training examples X_j in the current mini-batch. This is repeated with another random mini-batch until we've gone through all the training inputs (this is called an epoch).

VI. Testing and Validation

Scikit-learn libraries were used for forming predictions for the first two parts of the experiment. Part 3 required creating a custom script to input test data into the network and counting the number of accurately determined images. On a side-note, the misclassified images were also captured and considered for improving the predictions. In certain cases, they were used to improve the data in corresponding misclassified categories and provided up to a 5% increase in accuracy from the initial prediction attempt to the final prediction attempt.

The retraining was run for $4k$, $8k$ and $32k$ iterations. Figures 3 and 4 show the results from training the network for $32k$ iterations. The orange line shows the training accuracy and cross-entropy respectively in either figure. The figures show that the accuracies and the cross-entropy deviate after approximately 3200 iterations. This was due to over-fitting.

The accuracy was calculated by simply counting the number of truly classified images. Cross-entropy is more abstract. The cross-entropy is defined by the following formula [13]. The cross entropy is a metric to represent the loss in learning from training a neural network.

$$C = -\frac{1}{n}\sum_{x}[ylna + (1-y)ln(1-a)]$$

Where:

- n: total number of items of training data
- x: training input
- y is the desired output

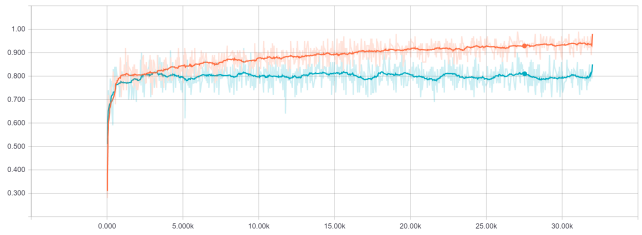

Fig. 3. Inception v3 CNN Retrain Accuracy

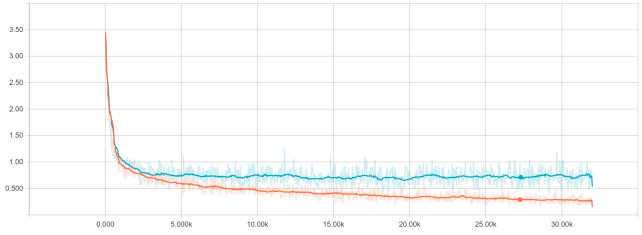

Fig. 4. Inception v3 CNN Retraining Cross Entropy

A. Logistic Regression

The logistic regression classifier was able to achieve accuracy scores in the range of 20% to 22%. This used the standard scikit-learn library. An attempt was made to increase the accuracy using principal component analysis (PCA) however that attempt did not prove fruitful.

B. Feed-forward Neural Network with 3 layers

The feed-forward neural network was tested using cross-validation. Due to it's simple architecture, the network was only able to achieve accuracy scores ranging between 26% and 34% on 5-fold cross validation. Since this performance was low, we opted to not run it against the public leaderboard.

C. Transfer Learning with Inception v3 CNN

The Inception v3 pre-trained convolutional Neural Network (pictured in fig. 9 – 12) achieves consistent accuracy scores ranging between 89.758% and 91.21% on the public leaderboard. In order to achieve maximal test-set prediction accuracy several techniques were tried: image perturbation, training set augmentation, and hyper-parameter cross-validation, as described in Table I.

Image perturbation such as random cropping, scaling, flipping, did not significantly impact prediction accuracy 90.545% vs 90.242%. During the cross-validation phase several perturbation intensities, and iteration values were used 5%, 10% and 15%, and 4k, 8k, 32k iterations, respectively. This did not result in any major accuracy gains, but did result in increased training execution time in the order of weeks. Indeed CNNs that use random Dropout, such as Inception v3, are not only computationally efficient, but also extremely robust to image perturbations, and avoid over-fitting [7], therefore do not require this type of image pre-processing.

Training set augmentation with additional images from the ImageNet public dataset similarly did not result in significantly increase accuracy (90.545% vs. 91.212% on public test set). This indicates the current number of training examples is sufficient. Interestingly enough, the submission that achieved a 91.212% accuracy score on the public test set achieved a lower score of 90.879% on the private test set, which might be symptomatic of cross-contamination between the test set and training set.

Hyper-parameter optimization through cross-validation has resulted in the largest accuracy gain (89.756% with 8000 iterations vs. 90.545% with 32000 iterations). Several other values were attempted to boost the accuracy however this approach did not provide substantial results.

VII. Discussion

A. Class Imbalance

Some issues were identified with the dataset that are worthy of mention. First and foremost, there is a clear class imbalance problem in the data. Figure 5 demonstrates the extent of

TABLE I
METHODS USED AND ACCURACIES OBTAINED [TEST-SET]

Method	Accuracy
Random	2.50%
Logistic Regression	20.23%
All zero	30.485%
CNN 4k	89.758%
CNN 8k	90.242%
CNN 32k	90.545%
CNN 8k: random crop, scale, brightness, flip left right	90.545%
CNN 30k: resize nearest neighbor, random crop, scale, brightness, hue, flipping, image standardization	90.182%
CNN 32k: training set augmentation	91.212%

the imbalance. As mentioned in table I, guessing label 0 would yield an accuracy of 30.485% which beat the random prediction. Ideally the classes would be balanced but we did not attempt to rebalance the data.

Fig. 5. Training set per class image distribution

B. Image Pixelation

The dataset contained many images which were not detailed enough to clearly distinguish by eye. This was seen as a problem because it would be much harder for inferring labels. Figure 6 shows a pixelated image that shows a dog but may be otherwise difficult to infer based on training data.

C. Foreground Images

Some of the images contain more than one foreground subject which can cause a misclassification since the subject of the image may be chosen incorrectly or an image belongs to two classes simultaneously.

D. Class Similarity

Relationships between labels were not fully exploited. Classes 20-22 have different types of squash which should have normally been handled based on an inferred genial structure for the species of squash and overarching parent species. This was not done. The labels were frequently misclassified by the training algorithm. Figures 7 and 8 show different types

Fig. 6. Misclassified Dog

of squash that appear similar in shape, but not in color. This also caused some images to be misclassified. Some classes are similar in nature and will inherently not always be handled well.

Fig. 7. Squash 1

VIII. CONCLUSION AND FUTURE WORK

Of the three methods considered in this report, convolutional neural networks performed the best. The other methods did not produce significantly relevant results to the experiment.

Fig. 8. Squash 2

Convolutional neural networks have been shown to be the method of choice for image recognition tasks. Their strength in image recognition comes from their ability to leverage the benefits of using convolutions on images for shape detection.

There are multiple opportunities to extend convolutional neural nets on the tiny ImageNet task. Namely, one of the most promising is leveraging ensemble methods. In fact, a ensemble method of 6 different classifiers led to the highest accuracy in the 2016 ImageNet challenge [14]. A future experiment may have an ensemble of several different classifiers specialized for expert classification such as different types of squash. Ideally, each classifier would contain classes for its specialization and a negative class for the rest of the images. Images falling into negative classes could be fed into other classifiers in the ensemble.

IX. Statement of Contributions

- Michael, Sorin and Srinivas all helped in writing this report. Michael and Sorin took point on building the convolutional neural network system, with Michael also working on the baseline classifier. Srinivas implemented the gradient descent and feed-forward neural network used for this project.

"We hereby state that all the work presented in this report is that of the authors."

References

[1] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[2] Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015).

[3] Szegedy, Christian, et al. "Rethinking the inception architecture for computer vision." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016).

[4] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016).

[5] Szegedy, Christian, et al. "Inception-v4, inception-resnet and the impact of residual connections on learning." arXiv preprint arXiv:1602.07261 (2016).

[6] Krizhevsky, Alex, and Geoffrey Hinton. "Learning multiple layers of features from tiny images." (2009).

[7] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." Journal of Machine Learning Research 15.1 (2014): 1929-1958.

[8] Zeiler, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks." European conference on computer vision. Springer International Publishing, (2014).

[9] Bahrampour, Soheil, et al. "Comparative study of deep learning software frameworks." arXiv preprint arXiv:1511.06435 (2015).

[10] Donahue, Jeff, et al. "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition." Icml. Vol. 32. (2014).

[11] Tensorflow. "How to Retrain Inception's Final Layer for New Categories — TensorFlow." TensorFlow. Tensorflow, 8 Mar. 2017. Web. 16 Mar. (2017).

[12] Nielsen, Michael A. "Using neural nets to recognize handwritten digits." Neural Networks and Deep Learning. Determination Press, 19 Jan. 2017. Web. 16 Mar. 2017.

[13] Nielsen, Michael A. "Introducing the cross-entropy cost function." Neural Networks and Deep Learning. Determination Press, 19 Jan. 2017. Web. 16 Mar. 2017.

[14] Lee, Purushwalkam, Cogswell, Crandall, Batra. "Why M Heads are Better than One: Training a Diverse Ensemble of Deep Networks." Web. 16 Mar. (2017).

Appendix

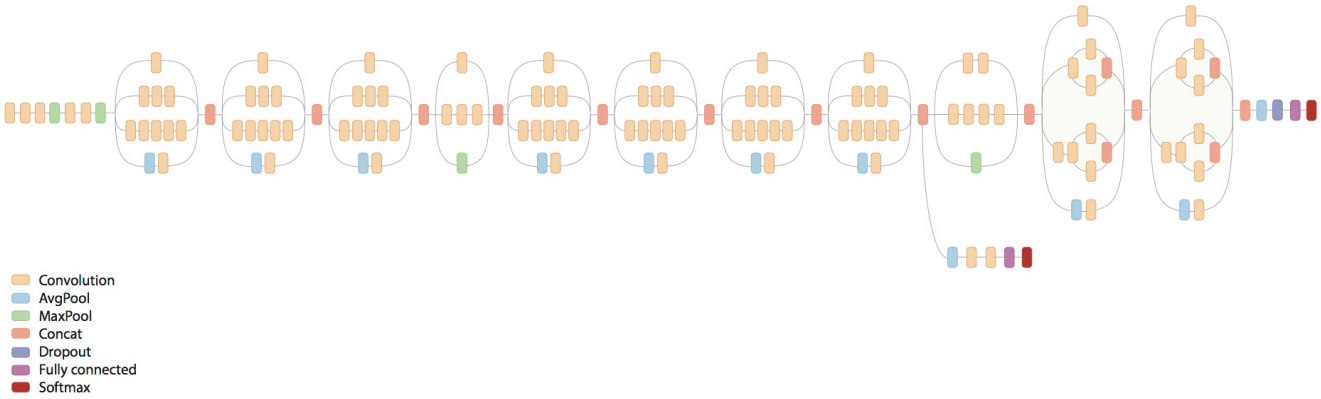

Fig. 9. Architecture of Retrained Inception v3 Model

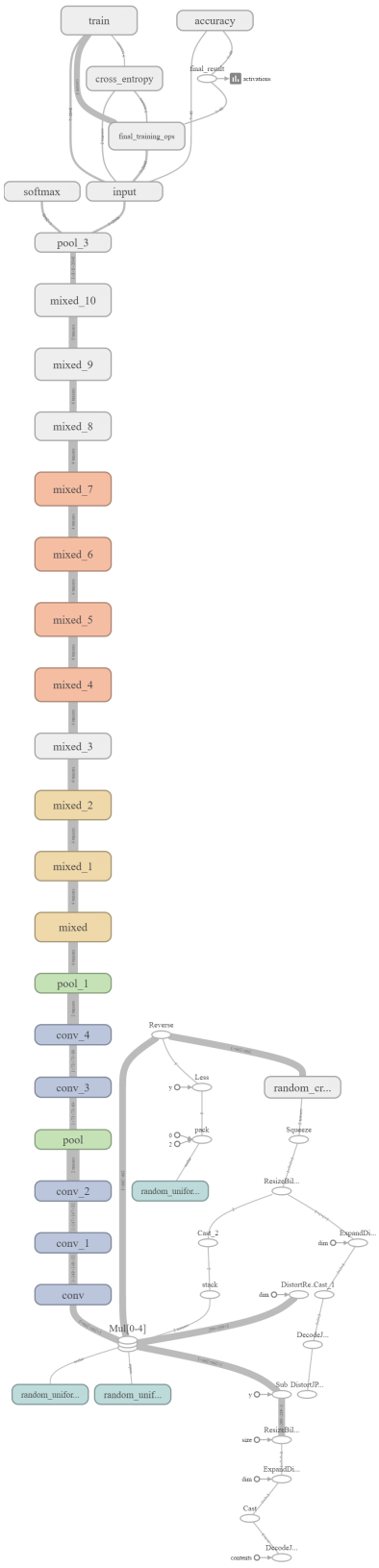

Fig. 10. Architecture of Retrained Inception v3 Model

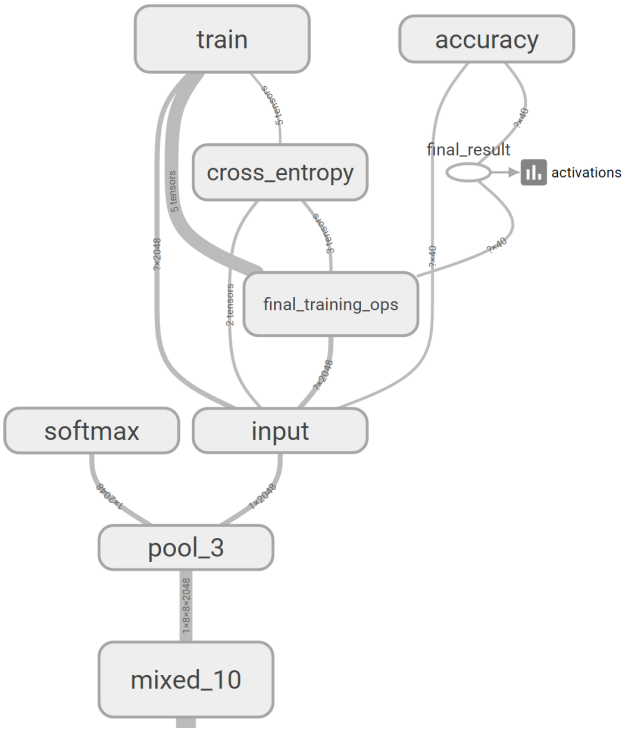

Fig. 11. Upper Retrained Inception v3 Model

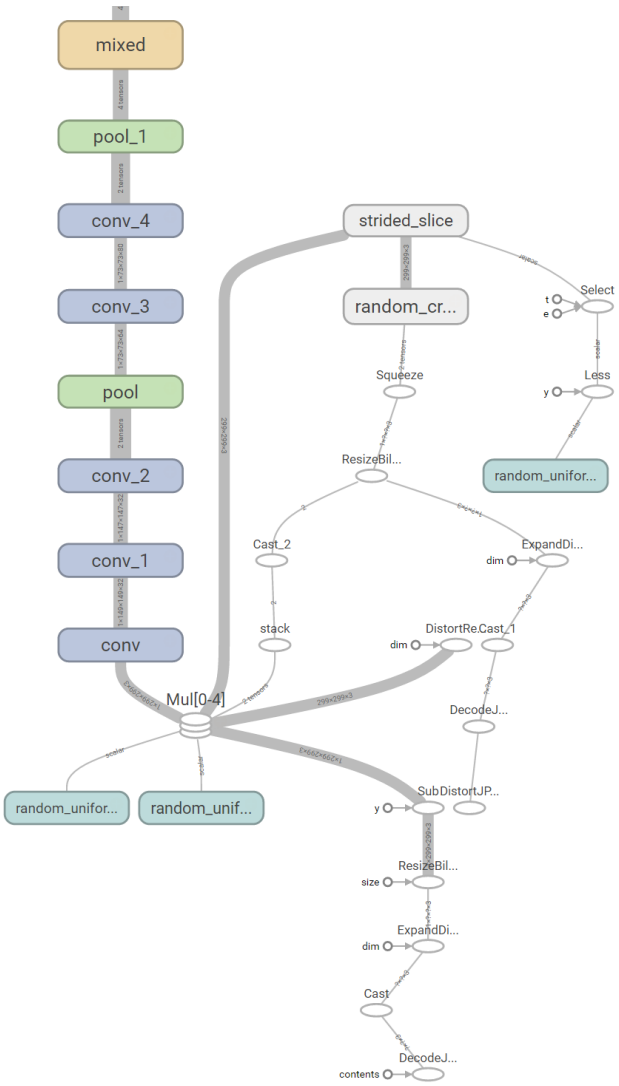

Fig. 12. Lower Retrained Inception v3 Model