"PawSitive": Cats vs Dogs Classification Using Convoluted Neural Networks and Deep Learning (April 2017)

Kaggle: PawSitive

Evan Knox (McGill University, 260502120, evan.knox@mail.mcgill.ca)
Kristine A. Juhl (McGill University, 260760431, kristine.juhl@mail.mcgill.ca)

***Abstract*—This report outlines how we implemented a Convoluted Neural Network (CNN), applying deep learning principles to classify images of Cats and Dogs. We implemented the CNN using the TensorFlow [1] library and image processing library OpenCV [2]. We achieved a success rate of 100% (0 misclassified images) using our methodology outlined in the paper.**

I. INTRODUCTION

Classification of images is one of the many new problems tasked to engineers and computer scientists in the 21st century. Motivations include medical imaging, sports, GIS and a wide range of other fields. The motivation of our project is to find a computational method to correctly classify images of cats and dogs.

Figure 1 Common images of household cat [11] and dog

To do this, we used a Convoluted Neural Network, which we implemented through Google's TensorFlow [1] open source library. Our approach was as follows:

1. Search through the literature to find the best possible algorithm for image classification.
2. Acquire a large training set of known images of Cats and dogs
3. Decide on best feature representation based on the state-of-the art algortihms
4. Feed the training set of images to the convoluted neural network
5. After n iterations, save the network and weights
6. Run the test set of images on the network and measure the performance.

II. PROBLEM REPRESENTATION: FEATURE DESIGN AND SELECTION

Up until 2012 the common approach for image classification was based on a "bag-of-features"-approach [3]. As images are represented by a great number of pixels, it can be important to find an alternative representation of the images that captures the important features of the image.

As an initial test for this project, the SIFT-descriptors [4] and Local Binary Patterns (LBP) [5] was clustered into 50 bins each using a K-Means algorithm and thereby creating a binary feature matrix. These features capture local image features and texture of image areas, respectively, and have both been found useful for image classification. Other possible features include Harris Corners [6], Difference-of-Gaussians (DoG) [7], Histograms-of-Oriented-Gradients (HoG) [8] and many more.
Based on preliminary results on these it was seen that SIFT-features alone could give a reasonable accuracy and the accuracy improved slightly when additional features were included.

Instead of searching to optimize the features used for representing the images, it was chosen to move towards the resent state of the art algorithms, that promised better

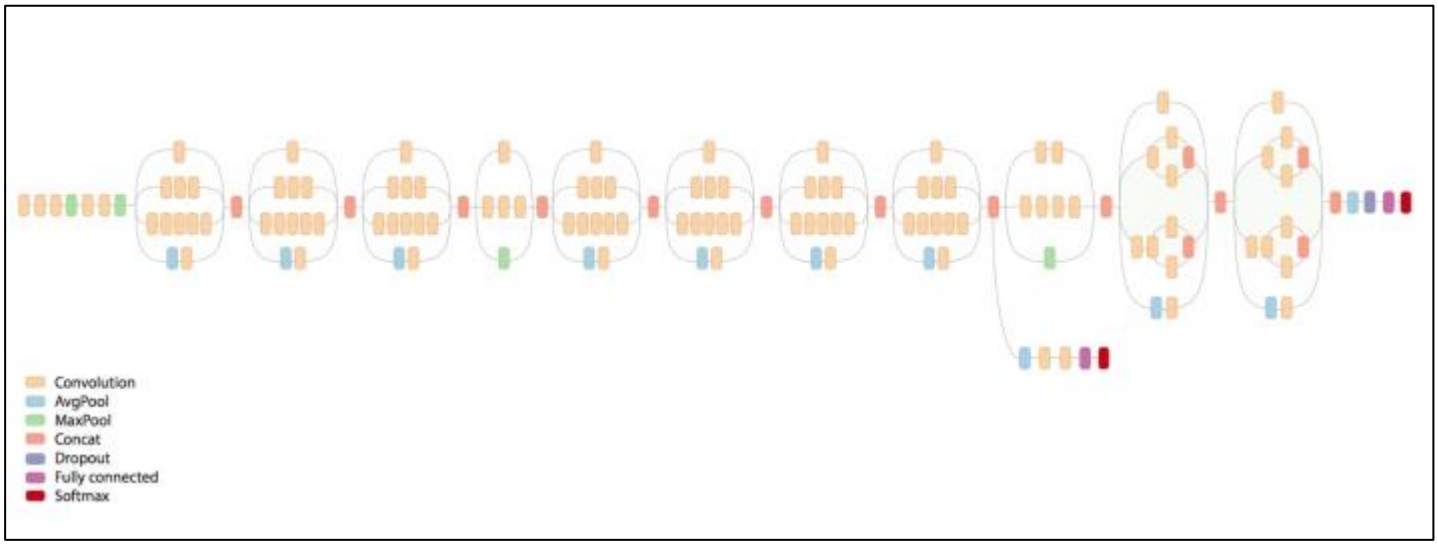

Figure 2 Schematic Diagram of the Inception-V3

results. After 2012 the common approach for image classification drastically changed after Convolutional Neural Networks were introduced to the field [9]. The CNN takes the raw pixel values in RGB space as input and automatically finds higher order features based on the architecture and weights in the image. One can think of the first many layers of the network as an automatic feature extraction stage, which is fed into a classifier in the end.

The Inception V3 Network as visualized in Figure 2 consists of a series of inception modules. These modules make it possible to combine features on different levels by concatenating the output of different sized kernel-convolutions and pooling functions. By combining multiple convolutional modules the final features fed into the softmax classifier becomes very complex, but are found optimal for the task.

III. Algorithm Selection and Implementation

After researching the success of various classification methodologies (KNN, SVM and CNN) we determined that a Convoluted Neural Network (CNN) [10], would yield the greatest overall success on the project. As seen in the figure below, the depth of the CNN is also crucial for classification accuracy.

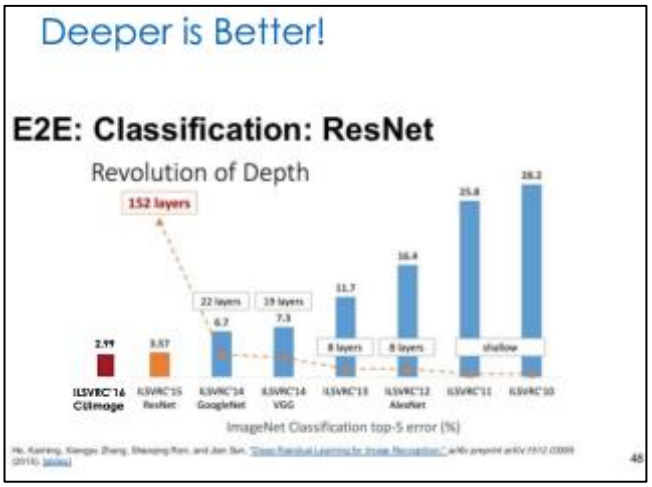

Figure 3 A slide from McGill Universities ECSE 415 CV class on the performance of various classifiers

Unfortunately, depth also corresponds to long computation times. The issue with a CNN is that the larger it is the more computationally expensive it is to train them. As CNNs typically have a large network with many weights to train, training from scratch can take many months dependent on the available computational power. Furthermore, the provided dataset is small and training such a large network based on a small training set increases the risk of overfitting. To have a deep classifier that is trainable in a reasonable amount of time, we applied the concept of transfer learning.

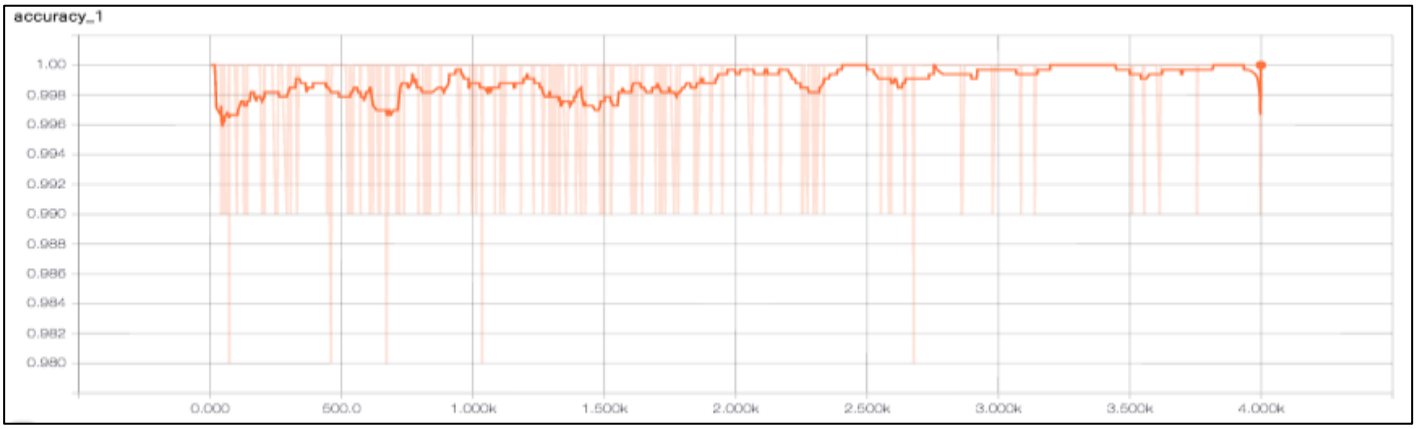

Figure 4 The accuracy of the CNN for each epoch of training

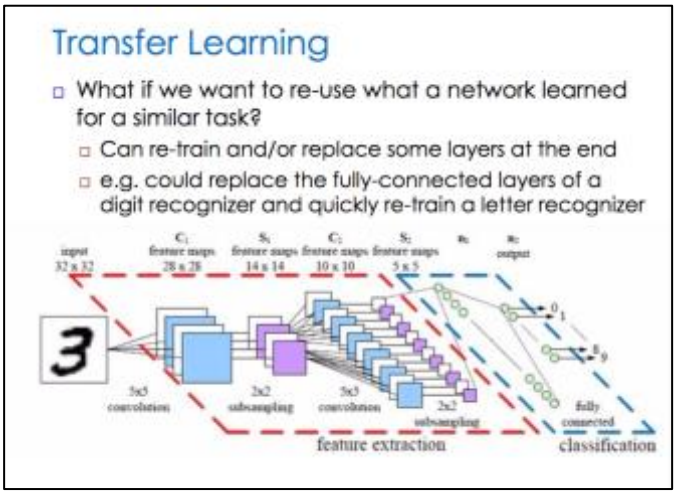

Figure 5 A slide from McGill Universities ECSE 415 CV class on transfer learning

As these problems are common in the field of Machine Learning Google decided to release a library and code for repurposing the Inception-V3 model yourself. Among many other features this library allows the user to do transfer learning – retraining or fine-tuning the network to the classification task at hand. The used Inception V3 model as visualized in Figure 2 makes use of a fully connected NN layer followed by a simple soft-max function to make the final predictions. This layer is specifically trained with the given cats and dogs training-set.
Before this a layer of dropout is added, this layer is important to reduce overfitting.

The process is as follows, and is simple enough to implement in TensorFlow.

1. Segregate the images of Cats and Dogs in our training set folder to a /Dogs/ and /Cats/ folder.
2. Load in the weights of an already trained InceptionV3 network. Doesn't matter what images the initially trained network was trained on. As long as it was on images and the old network was classifying them.
3. Strip the final layer (Fully connected network and soft-max function) of the old network, and replace it with a new fully connected NN and softmax function with two possible outcomes (Cats and Dogs).
4. Retrain the network in relatively quick time (around 10 minutes)
5. Run the test set through the network and check to see the outputs.

IV. Testing

Conveniently, TensorFlow provides a program called TensorBoard, which allows you to visualize the training of the Network over time.

As you can see from Figure 5 the CNN we trained isn't 100% accurate, rather it has an accuracy of around 99.5% - 100%. It's still undoubtedly impressive however, and as we ran our network over more epochs of the training data set, the networks accuracy improved to nearly 100%. We additionally tested on various 'Bad Images' to see how the network would react to them. The results were impressive. Given an image of a girl and a cat, it would classify cat. Given an image of 2 dogs it would also classify dog.

As for the Kaggle results, we were first place from our very first submission, receiving a score of 100%. Retraining the network again and running the test set through the network also produced a score of 100%.

V. Discussion

The above results shows what a strong tool Deep Neural Networks can be in the task of image classification. Instead of hand tuning each feature the network itself learns the most useful features based directly on the provided data. This removes possible noise in the features introduced based on ie. specifying the wrong number of clusters for the bag-of-features approach or choosing features without any information useful for the task.
By using a DNN all weights are updated simultaneously which makes all intermediate states optimized jointly. This ensures that each intermediate representation not only contains as much information as possible, but also that the information contained in different features do not overlap unnecessarily.

On the downside, using a DNN greatly reduces the interpretability of the process and it is impossible to infer any information on what distinguishes a cat from a dog. This also makes it hard to optimize the network as the intuitive understanding of the feature extraction is removed. Special for the Inception-networks is the use of the inception-modules. This greatly reduces the number of parameters in the network and thereby the number of operations necessary, but also reduces the interpretability.

As our first submission, we were both very surprised to see that a perfect classification was obtained using this model. Because of this it has not been possible to investigate if better results could be obtained by using other techniques. If more complex data sets were at hand it might be necessary to further optimize the algorithms.

One reason that this classification algorithm does very well, could be that the provided data-set is quite homogeneous and the test-images look very like the training-images. In cases where this may not be the case, it might prove useful to introduce more variations of the already known training images. This could easily be done by randomly change the brightness, contrast and angle of rotation ($\pm 45°$) of the images.
Another possible addition would be to segment the image into animal and background, and crop away the background. This method has potential for removing some unnecessary information about the background. This method may have a slight improvement of the results, especially if a lot is going on in the background. Cropping away the background do however also take some information away, as the environment where we often find cats and dogs may be different. This environmental information is thus taken away.

VI. References

[1] Google Inc, "TensorFlow: Large-scale machine learning on heterogeneous systems,," 2015. [Online]. Available: http://tensorflow.org/. [Accessed 30 04 2017].

[2] G. Bradski, "opencv library," *Dr. Dobb's Journal of Software Tools*, 15 01 2008.

[3] C. R. D. L. F. J. W. a. C. B. Gabriella Csurka, *Visual Categorization with Bags of Keypoints*, 2008.

[4] D. G. Lowe, "Object recognition from local scale-invariant features," *International Conference on Computer Vision*, pp. 1150-1157, 1999.

[5] L. W. a. D. He, "Texture Classification Using Texture Spectrum," *Pattern Recognition*, vol. 23, no. 8, pp. 905-910, 1990.

[6] C. H. a. M. Stephens, "A combined corner and edge detector," *4th Alvey Vision Conference*, pp. 147-151, 1988.

[7] a. E. H. D. Marr, "Theory of Edge Detection," *Royal Society of London. Series B, Biological Sciences 207.1167*, pp. 187-217, 1980.

[8] N. D. a. B. Triggs, "Histograms of Oriented Gradients for Human Detection," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 886-893, 2005.

[9] M. T. Review, *The Revolutionary Technique That Quietly Changed Machine Vision Forever*, ttps://www.technologyreview.com/s/530561/the-revolutionary-technique-that-quietly-changed-machine-vision-forever/, 2014.

[10] Google Inc., "Going Deeper with Convolutions," *CvF*, 2015.

[11] E. Knox, *Image of Gatsby*, Ottawa, Ontario, 2016.