Kaggle Competition Dogs vs Cats

1) DEFINITION

a) ***Project Overview***

In this competition, the goal is to use a machine learning algorithm to classify whether images contain either a dog or a cat. If we look at this kind of problem from a high- level point of view, we would find that while a simple task like this one is easy for humans, on the contrary, it is very difficult for a computer.

b) ***Problem Statement***

Web services are often protected with a challenge that's supposed to be easy for people to solve but difficult for computers. It is called CAPTCHA (Completely Automated Public Turing test to tell Computers and humans Apart). CAPTCHAs are used for many purposes, such as to reduce email and blog spam and prevent brute force attacks on website passwords.

One type of CAPTCHA is called Asirra(Animal Species Image Recognition for Restricting Access), it works by asking people to identify photographs of cats and dogs which is an easy and fun task for humans and a very hard one for computers. Asirra is unique because of its partnership with Petfinder which is the world's largest website devoted to finding homes for homeless pet. They provided over three million images of cats and dogs manually classified by people. For the sake of this competition, we will only use a subset of these images. An example of Asirra is shown in figure 1.

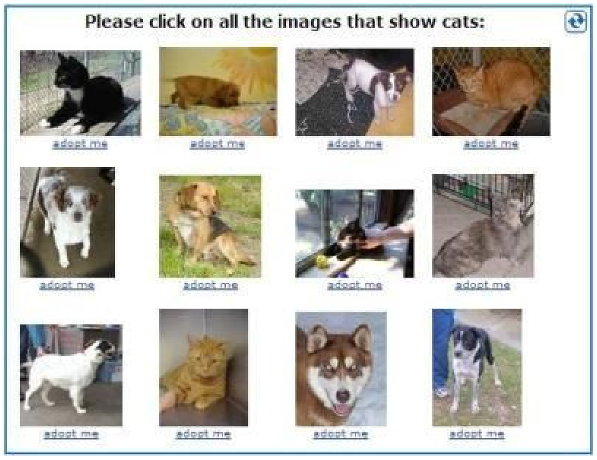


Figure 1, Asirra

The purpose behind this competition is to explore how well a computer can get in solving Asirra using the most recent Machine learning algorithms. One of the ways that a computer can solve this problem is by random guessing the images in Asirra, however, by using machine learning, a computer can go a step further where it can make a guess that is far from random but based on knowledge and understanding of the context of those images by learning about a wide diversity of cats and dogs images in terms of different backgrounds, angles, poses, lighting, etc.

We will attempt to solve this problem using Deep learning or more specifically, Deep Residual learning, this method is a good fit to solve this problem because it will allow us to train our algorithm using very deep convolutional neural networks (i.e consist of many layers). As the neural network gets deeper, the more it can learn about diverse representations of an image. The idea is that very deep convolutional neural networks are hard to train, however, it was proven that the use of a residual learning framework can ease the training of very deep networks. The trick is that the layers of the network are reformulated as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. The result is that deeper convolutional neural networks are easier to optimize and at the same time can gain accuracy from increased depth.

c) <u>Metrics</u>

The performance of this problem will be measured based on the log loss function:

$$\textbf{Log loss} = \frac{-1}{n}\sum_{i=1}^{n}[y_i \log(\mathbb{y}_i) + (1 - y_i)\log(1 - \mathbb{y}_i)]$$

n = Number of images in the test set
$\mathbb{y}_i$ = Predicted probability of the image being a dog
$\boldsymbol{y_i}$ = 1 if image is a dog and 0 if it is a cat
log( ) = natural (base e) logarithm

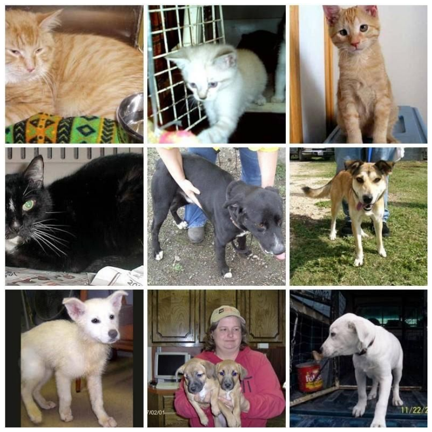
We could also use the Accuracy as a performance metric for this problem, where we simply count the predictions where the predicted value equals the actual value but this is not a good indicator because it just gives an absolute answer about the predictions. On the other hand, The Log Loss gives us further insight for our predictions as it takes into account the uncertainty of a prediction based on how much it varies from the actual label.

The log loss function will measure the performance of our classification model where the prediction input is a probability value between 0 and 1, and the goal of our machine learning algorithm will be to minimize this value. A high-performance model will have a log loss score closer to 0 and a smaller log loss means higher accuracy

2) ANALYSIS

a) ***Data Exploration***

The dataset consists of **37000 RGB images**, the **training set** consists of **25000 images**, **12500 for dogs** and **12500 for cats** with varying resolutions; a random sample of 10 images from the training set has the following sizes **[(500, 374), (499, 333), (360, 485), (377, 500), (500, 395), (499, 375), (500, 374), (342, 500), (499, 333), (500, 374)]** and the **test set** consists of **12000 images** mixed between cats and dogs. As shown in figure 1, The images captured are very diverse as they include dogs and cats in different **poses**, **lighting** and **backgrounds, figure 2** shows examples of images from the training set & a barchart showing the class distributions



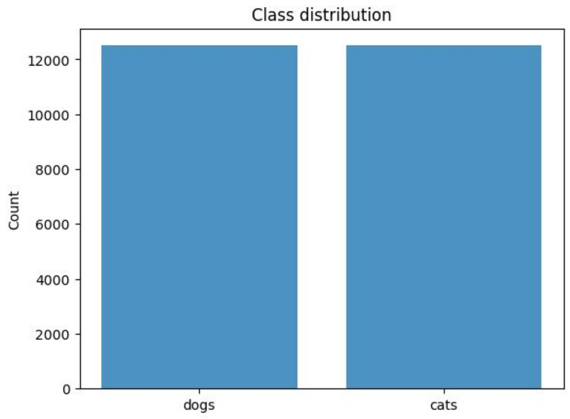


Figure 2, examples from the training set & Class distribution barchart

b) ***Algorithms and Techniques***

The algorithm that will be used for this project is deep residual learning , this one is a good fit for this problem because it enables us to easily train a convolutional neural network with many layers and therefore will learn many different representations of dogs and cats and consequently increase the output classification rate. The core idea of the algorithm is reformulating the neural network layers as learning residual functions with reference to layer inputs instead of learning unreferenced functions.

Increasing the number of layers in a neural network does not lead to a better performance due to the vanishing gradient problem which interrupts convergence. This problem was solved using normalized initialization and intermediate normalization layers which enable deep networks to converge when trained with Stochastic gradient descent and back propagation.

However, as the network gets deeper, it introduces a degradation problem, whereas the network converges, its accuracy gets saturated and degrades rapidly, as shown in figure 3, the deep 56-layer neural network has a higher training error and consequently test error when compared to the smaller 20-layer network.

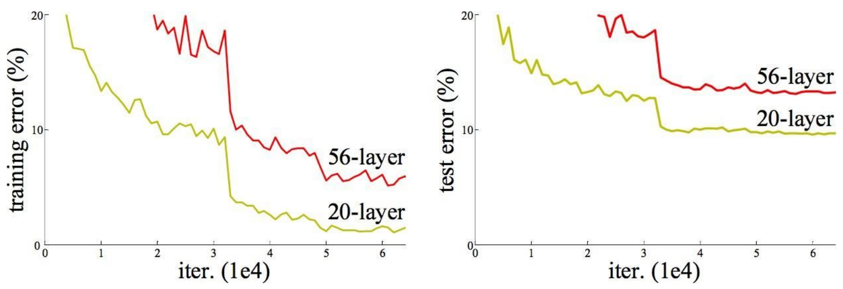


Figure 3, the degradation problem

The design of deep residual networks depends on two concepts, Residual representations and shortcut connections. Residual representation are feature representations that depend on the idea of characterizing a signal with a gradient vector derived from a generative probability model and to subsequently feed this representation to a discriminative classifier. In the application of image categorization where the input signals are images and the underlying generative model is a visual vocabulary: a Gaussian mixture model which approximates the distribution of low-level features in images. On the other hand, Shortcut connections are connections that skip one or more layers in the network.

Therefore, combining residual learning and shortcut connections gives us residual learning where instead of expecting each few stacked layers to fit a desired underlying mapping, they instead fit a residual mapping. If **H(x)** is the desired underlying mapping where **x** is the inputs to these layers, we try to make the stacked non-linear layers fit another mapping **F(x) = H(x) – x** , then the original mapping becomes **F(x) + x as shown in figure 4**

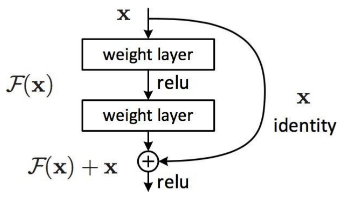

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 4, Residual Learning, building block

c) ***Benchmark***

The performance of our model will be assessed based on the position of our score in the Kaggle public leaderboard for the project. We will aim to be within the top **25%** of the 1314 teams in the competition leaderboard.

3) METHODOLOGY

a) Data Preprocessing

Each image is resized with its shorter side randomly sampled in [256, 480] for scale augmentation. In addition, a 224×224 crop is randomly sampled from the image or its horizontal flip, with the per-pixel mean subtracted and standard color augmentation is used. These steps are very important because we want our neural network to learn an invariant representation of the images, this includes scale invariance (where the neural network does not change its prediction if the size of the object in the image changes), rotation invariance (we do not want the angle of the object to matter in the image or translation invariance (The network does not care if the object is on the left side of the image or the right side of the image, it will still identify it as a dog).

b) **Implementation**

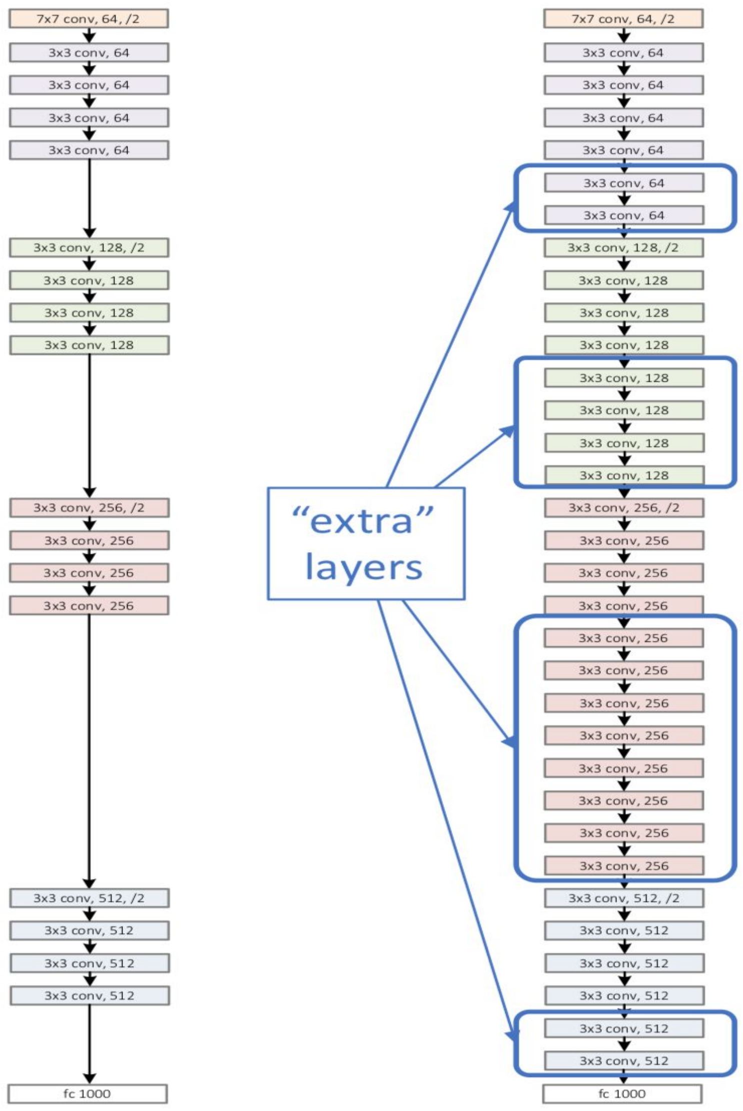
The algorithm was implemented in python using the following libraries:
Torch, **Torch vision**, **Python Imaging Library** on **4 Nvidia Titan x GPUS** which have
3072 cores each. We used **Stochastic Gradient Descent** with a mini-batch size of **256**. The learning rate was **0.006**, a weight decay of 0.0001 was used and the momentum is **0.9**, In the first implementation, the model that we used was **resnet18**.
For each of the two classes we used **10500** images for **training** and **2000** images for validation. The implementation is written in four .py files, model, metrics, Train and Test, the implementation of the algorithm was as follows:

1) **model.py**
   Used for downloading a pre-trained resnet18 that was already trained on the ImageNet dataset, This is available on the Torchvision website The model has 1000 output classes, so to make it suitable for our case we use the same weights of the model and just convert the final layer from 1000 to 2 classes
2) **metrics.py**
   Used for creating the confusion matrix and calculating the accuracy, precision and recall
3) **Train.py**
   Used for doing the actual training on our dataset, the network takes a 224x224 image patch. The network uses the cross-entropy loss function and the stochastic gradient descent algorithm
4) **Test.py**
   Used for testing the network, it takes 5 patches of the input image (4 corners and 1 center) and it computes the output for all these 5 patches and takes the average of them as the final prediction for that image, we used 5 patches so that network is able to see the full image as they are not perfectly 224x224 resolution images

Most of the problems that were faced during the coding process were mainly related to figuring out how the torch vision library works, however, working with Keras and Tensorflow in the deep learning projects helped in quickly learning and understanding how to use TorchVision to train and test the model on a GPU

c) <u>Refinement</u>

The improvement that we attempted on this model is using a much deeper network, which means replacing resent18 with resnet34, figure 5, shows the difference between the two models and the extra layers that were added



4) RESULTS

a) <u>Model Evaluation and Validation</u>

The results that we got are as follows, for **resnet18** the log loss function score was **0.07172.** On the other hand, for **resnet34** the log loss function score was **0.06985.**

b) <u>Justification</u>

For **resnet18** the log loss score puts us around position **180** in the leaderboard. On the other hand, for **resnet34** the log loss function score puts us around position **167**, it's not a huge improvement but it's the best we can do, because if we go higher than 34, the network will start overfitting, in addition it is within the top 25% of the teams in the leaderboard, which is the benchmark we set to assess the performance of our model.

5) CONCLUSION

Reflection

Although this might look like a simple problem at first yet, its importance is major, it shows how well a computer can get in behaving like a human being and maybe even surpassing us. Because it shows how it can easily learn from many different representations of dogs and cats the concept of what a dog is and what a cat is. Using deep learning shows us how a computer can learn higher level concepts by making guesses that are far from random but are rather based on knowledge about the context of what it is trying to learn. The difficulties that I had with this project which also taught me a lot where in learning the concept of residual neural networks and figuring out how to use a GPU to train and test my system, in addition to learning about new machine learning libraries. All in all it was a major learning experience for me.