Neural Networks Final Project:
Comparing different data pre-processing techniques for convolution neural networks

Krystian Wojcicki, 101001444
Michael Kuang, 101000485

COMP 4107, Fall 2018

1 Introduction

Natural Images is a dataset currently featured on *kaggle.com*[1]. The set is comprised of 6899 images from 8 distinct classes of airplane, car, cat, dog, flower, fruit, motorbike and person. Each class has a varying number of examples. We examined different techniques in pre-processing images and implemented them to analyze their effects on performance in a convolution neural network.

2 Problem Statement

This project attempts to address the problems: "Do different image resizing techniques make a difference to classification accuracy?" and "Do balanced datasets improve classification accuracy?".

3 Background

One problem we looked at is the Class Imbalance problem, and this is the problem in machine learning where we have great discrepancy in the number of examples between class data. In other words, there are far more examples of one class than another in a set of data. This is a problem because as we know, machine learning works best when we have a more uniform distribution of data. In this project, we examine the performance between two up-sampling techniques called Synthetic Minority Over sampling Technique (SMOTE) and ADAptive SYNthetic (ADASYN) as solutions to class imbalance.

Another problem we investigated is how image resizing can affect performance. Given a set of non-uniform sizes of images, we must resize them to some constant shape in order to feed examples into the network. We examined two methods; resize the image by scaling it, and crop or pad the image about the center. The problem with resizing an image is that we inevitably lose information as images are scaled down or add noise as they are scaled up, but we are able to retain the information more uniformly. Cropping the image will directly lose the information as we resize the image by cropping out the outer most part of the image, while padding the image with black pixels evenly about center will retain the image aspect ratio.

3.1 The Dataset

The *Natural Images* dataset contains 6899 distinct RGB images of varying sizes for 8 classes as described below:

- airplane: 727
- car: 968
- cat: 885
- dog: 702
- flower: 843
- fruit: 1000
- motorbike: 788
- person: 986

3.2 Machine Learning Libraries

Four libraries were used to implement our code, and create our neural network model: Tensorflow, scikit-learn, imbalanced-learn and OpenCV.

3.3 Methodology

3.3.1 Scaling vs Crop and Pad

We used OpenCV's resize method to scale the images down to or up to a specified size using nearest neighbour interpolation. The nearest neighbour interpolation is a point sampling algorithm that, rather than calculate the average or some weighting criteria, simply determines the nearest neighbouring pixel and assumes the intensity value of it. Using this method, it will upscale or downscale to a specified size. This method will add noise when we upscale or lose information when we downscale, and it is noted that the image will lose its original aspect ratio.

Resizing an image by crop and pad means that the image will retain it's image resolution, but we either crop or pad the image about the center to the specified size. So, if the height of the image needs to be smaller, we crop the top and bottom evenly. Conversely, if the height of the image needs to be larger, we evenly pad the top and bottom with black pixels.

To crop images, we simply used Python's built in array slicing on numpy arrays. Then, to pad or resize images, we used the following methods from the OpenCV library:

- cv2.copyMakeBorder: padded the borders of the image by a specified amount using pixel color value of 0
- cv2.resize: resizes a given image to a specified height and width using interpolation method INTER_NEAREST

3.3.2 Spatial Pyramid Pooling

CNNs require a fixed input image size such as 32x32, 64x64 or 128x128 as experimented with in this report. This requirement is human made and artificial while also potentially decreasing recognition accuracy due to loss of content while padding or distortion while resizing. The need for fixed size inputs is due to the fully-connected layer portion of a CNN, the convolution layers use a sliding window and operate correctly on any size input. However the fully-connected portion requires a fixed-size input. To get around this Kaiming et al [2] introduced spatial pyramid pooling (SPP) into CNNs, this pooling layer is added on top of the final convolution layer and generates fixed-length outputs regardless of the size of the feature maps given to it.

SPP builds on and improves the concept of Bag-of-Words (BoW), it does so but maintaining spatial information by pooling in local spatial bins. These spatial bins have sizes that are proportional to the image size meaning that the number of bins is fixed regardless of the image size.

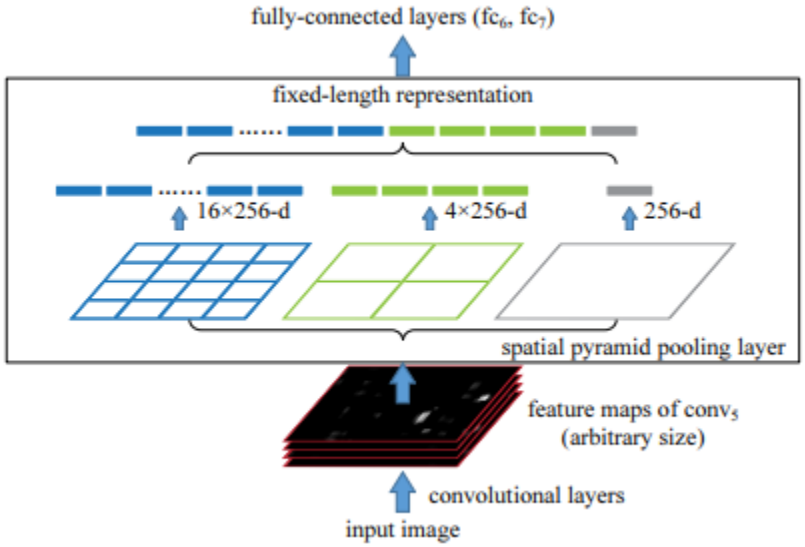

Figure 1: An example of a network structure using an SPP from Kaiming et al [2] where 256 is the number of input feature maps and the SPP consists of 3 different bin sizes: a global bin, a bin with 2 buckets and a bin with 4 buckets

3.3.3 Over-sampling vs Under-sampling

Over-sampling balances a dataset by randomly duplicating the "better" representations of elements from the minority classes. However, this may cause over fitting because of the duplications. Under-sampling balances a dataset by randomly eliminating elements uniformly from the majority classes as it makes a K-means to reduce the number of samples.

To over-sample our dataset, we used the following methods from the imbalance-learn library:

- imblearn.over_sampling.SMOTE: over-sampled on the image dataset using "not majority" for the *sample_strategy* parameter
- imblearn.over_sampling.ADASYN: over-sampled on the image dataset using "not majority" for the *sample_strategy* parameter

The SMOTE algorithm applies KNN approach where it selects K nearest neighbors, joins them and creates the synthetic samples in the space[3]. The algorithm takes the feature vectors and its nearest neighbors, computes the distance between these vectors, and multiplies it by a random number between (0, 1) and it is added back to feature. The ADASYN algorithm is based on the idea of adaptively generating minority data samples according to their distributions using K nearest neighbor[3]. The algorithm adaptively updates the distribution and there are no assumptions made for the underlying distribution of the data. The algorithm uses Euclidean distance for KNN Algorithm. The key difference between SMOTE and ADASYN is that the former generates the same number of synthetic samples for each of the minority classes while the latter uses a density distribution, as a criterion to automatically decide the number of synthetic samples that must be generated for each minority sample by adaptively changing the weights of the different minority samples to compensate for the skewed distributions.

3.3.4 Neural Network Model

The model chosen to experiment with was quite simple with 3 convolution layers, each with a kernel of 3x3 using a stride of 1x1 and padding of SAME outputting 32 feature maps. Max pooling was done after each convolution layer using the SAME padding with a stride of 2x2. These 3 layers connected to a fully connected feed forward network of 625 neurons. When using an SPP layer 3 bins were used of size 4, 6, 8.

3.3.5 t-SNE

t-SNE is another technique for dimensionality reduction and as described in the original paper[4], t-Distributed stochastic neighbor (t-SNE) minimizes the divergence between two distributions: one distribution which measures the similarities between inputs and the other distribution that measures the difference between the reduced data according to a t-distribution. The main drawback of this algorithm is that it scales quadratically so using more than a few thousand data point is unfeasible.

4 Results

4.0.1 Resizing unbalanced dataset by scaling using nearest interpolation

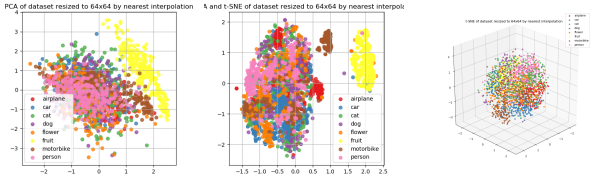

Figure 2: Visualization of 2000 randomly picked data points for 64x64 rescaled images

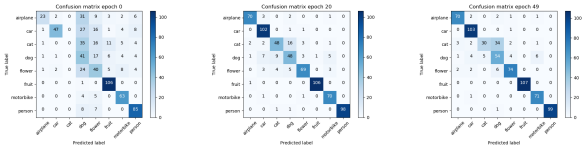

Figure 3: Confusion Matrices of unbalanced resized images to 64x64 by scaling

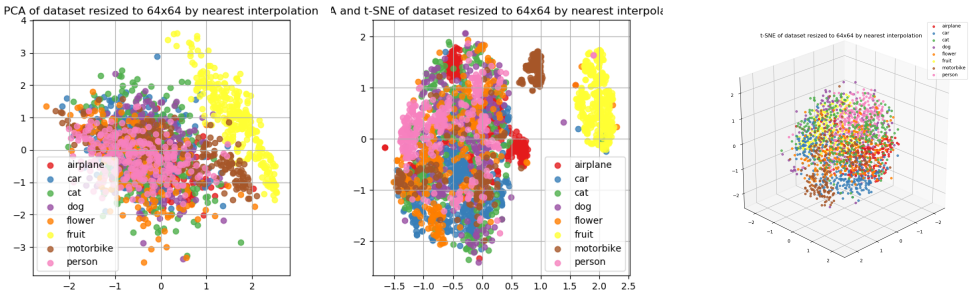

Figure 4: Visualization of 2000 randomly picked data points for 64x64 rescaled images

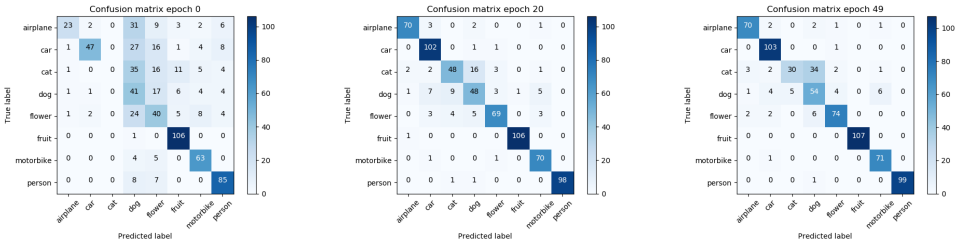

Figure 5: Confusion Matrices of unbalanced resized images to 64x64 by scaling

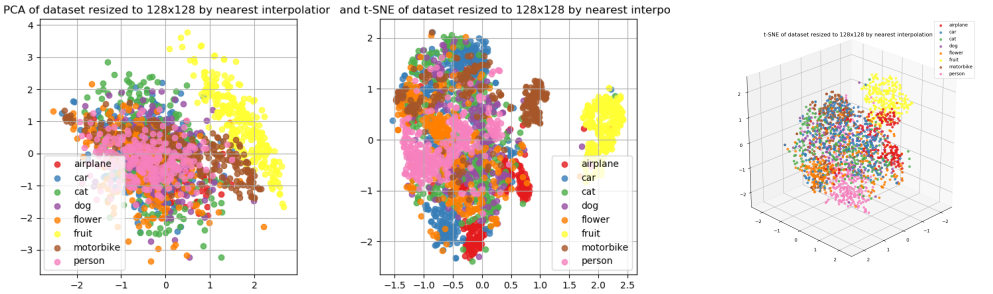

Figure 6: Visualization of 2000 randomly picked data points for 128x128 rescaled images

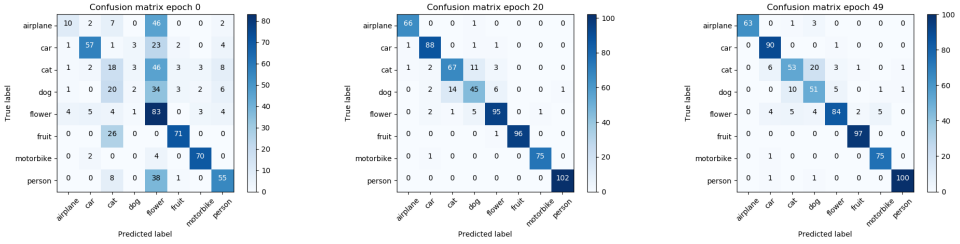

Figure 7: Confusion Matrices of unbalanced resized images to 128x128 by scaling

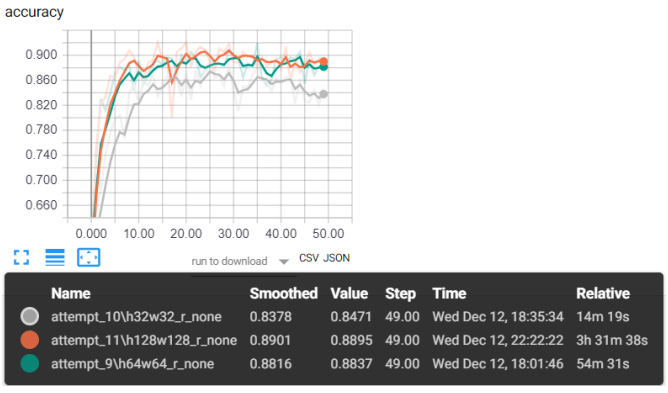

Figure 8: Testing accuracy comparison for image scaled to 32x32 (grey), 64x64 (green), 128x128 (orange)

4.0.2 Resizing unbalanced dataset by crop and pad

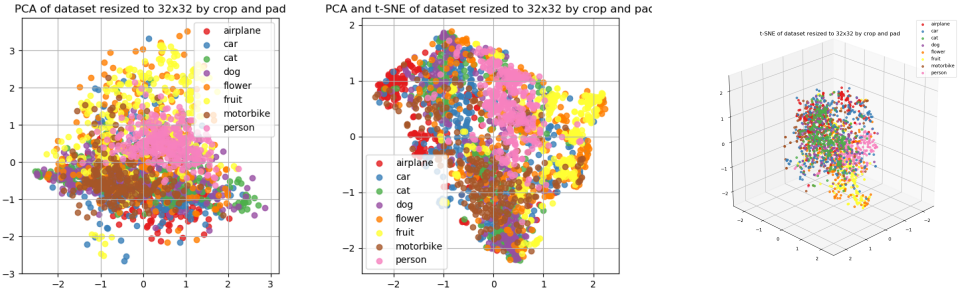

Figure 9: Visualization of 2000 randomly picked data points for 32x32 crop/pad images

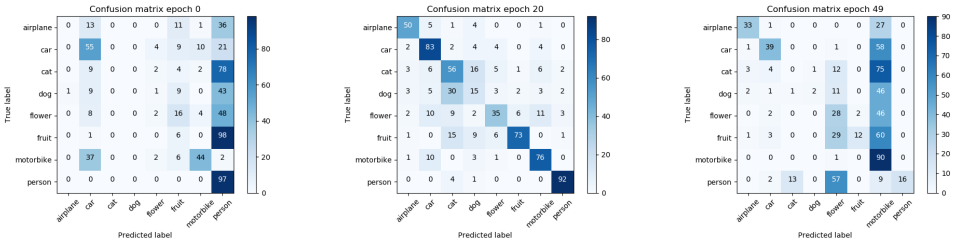

Figure 10: Confusion Matrices of unbalanced resized images to 32x32 by crop/pad

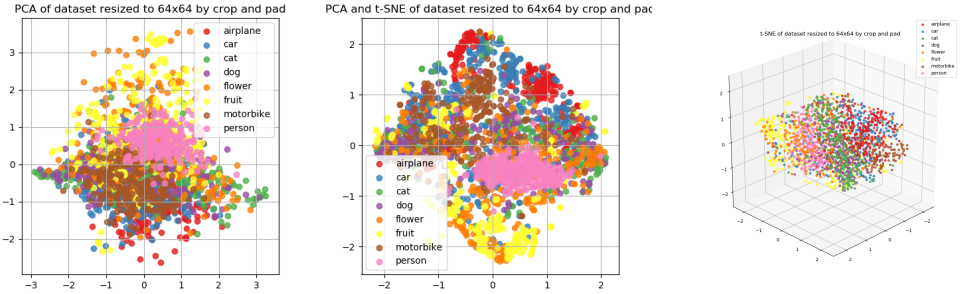

Figure 11: Visualization of 2000 randomly picked data points for 64x64 crop/pad images

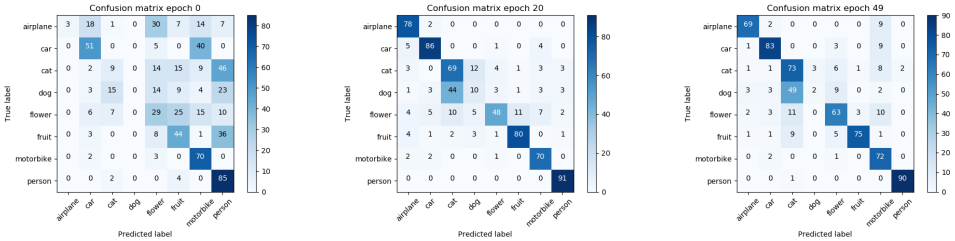

Figure 12: Confusion Matrices of unbalanced resized images to 64x64 by crop/pad

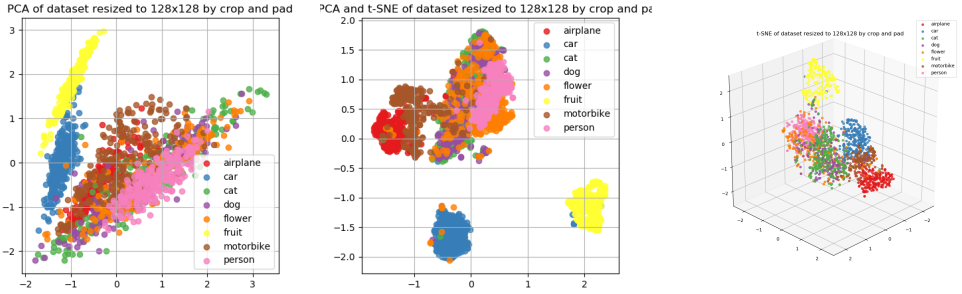

Figure 13: Visualization of 2000 randomly picked data points for 128x128 crop/pad images

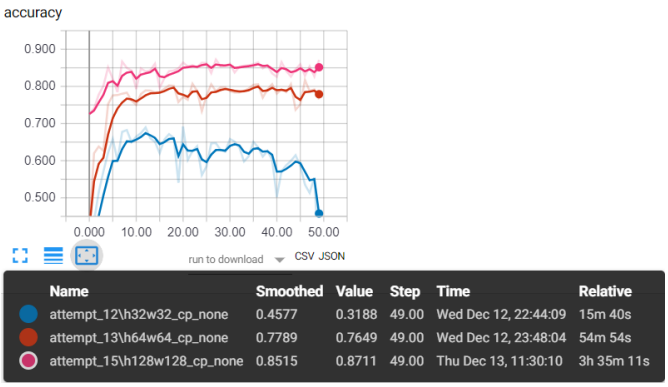

Figure 14: Testing accuracy comparison for image resized by nearest interpolation for 32x32 (blue), 64x64 (red), 128x128 (pink)

, ,

4.0.3 Up-sampling dataset using SMOTE and ADASYN on 64x64 rescaled images

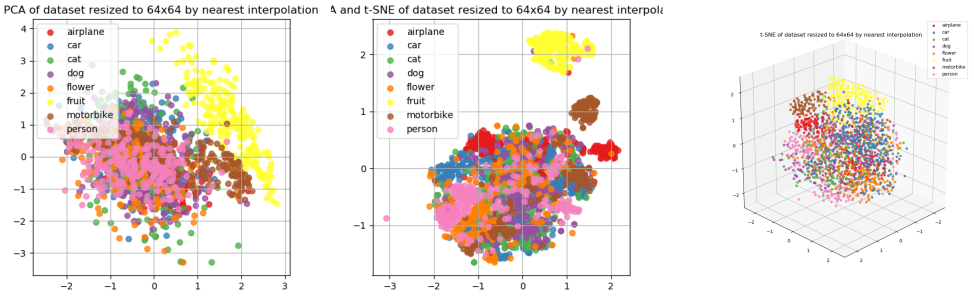

Figure 15: Visualization of 2000 randomly picked up-sampled data points by SMOTE for 64x64 images resized by nearest interpolation

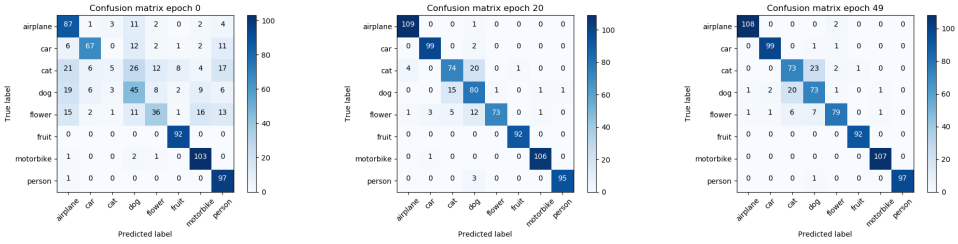

Figure 16: Confusion matrix up-sampled data points by SMOTE for 64x64 images resized by nearest interpolation

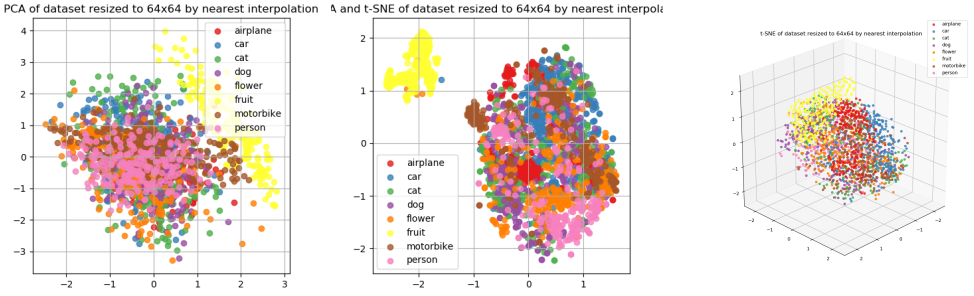

Figure 17: Visualization of 2000 randomly picked up-sampled data points by ADASYN for 64x64 images resized by nearest interpolation

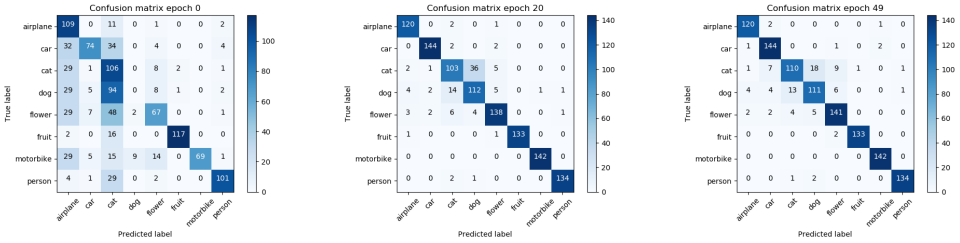

Figure 18: Visualization of 2000 randomly picked up-sampled data points by ADASYN for 64x64 images resized by nearest interpolation

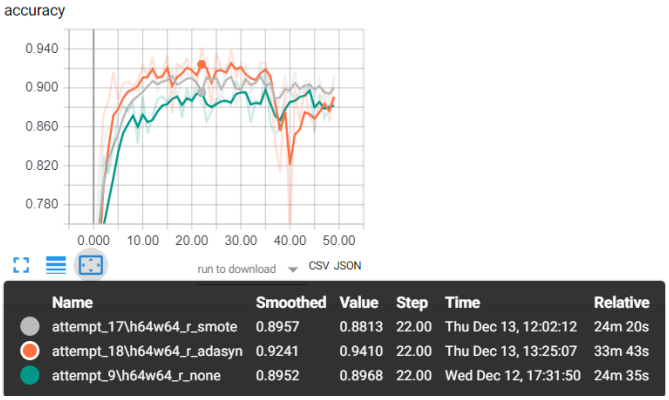

Figure 19: Testing accuracy comparison between no-upsampling (green), SMOTE (grey) and ADASYN (orange) on 64x64 image dataset

5 Analysis and Discussion

From the figures above we can see the data is highly intertwined and any clustering technique would have difficultly achieving a high accuracy. Using t-SNE on top of PCA was able to separate the data better than PCA on its own, again however it was not able to separate enough for any clustering technique to work.

We compared results between resizing by scaling and crop and pad. Each experiment ran for 50 epochs and the resulting accuracies were tabulated as shown below:

Table 1: Accuracy Comparison of Scaled vs Crop and Pad

Epoch	Scale			Crop and Pad		
	32x32	64x64	128x128	32x32	64x64	128x128
5	79.79	87.12	86.51	65.87	77.59	82.03
10	82.82	89.10	87.02	66.49	75.01	84.83
20	87.42	90.52	92.06	69.18	80.70	86.41
50	84.71	88.37	88.95	55.14	76.49	87.11

In general, the training seems to have converged at approximately 20 epochs before over fitting as we can see that at 50 epochs the accuracy is worse than at 20 epochs. The results from the different resizing techniques show that in general, resizing by scaling the image is significantly better than cropping and padding, as can be seen from Figure 8 and Figure 14 and Table 1. We note that each size increment for cropping and padding an image results in a significant accuracy increase and this is reasonable because of the nature of how crop and pad processes an image. When an image is resized by crop and pad, we either directly crop out the extra rows and columns of pixels if it is too large, or pad them with black pixels. The idea is to retain the image resolution and not skew it. However, if the original image is much larger than the resulting size, we directly remove a large amount of data which will cause poor performance. For example, if we resize a cat image of 64x64 to a 32x32 image by cropping it, the resulting image will only hold the center chest area of the cat, thus losing the "arm", "leg" and "head" features of a cat. Conversely, resizing an image by scaling it with nearest interpolation is a much better approach at reducing noise because each image still holds the full picture. This also explains why the images scaled to 32x32 does just

as well as an image that is cropped and padded to 128x128. However, we have not tried a large enough size in which cropping was unnecessary and the images would only be padded with black pixels. It may be interesting to observe the performance for images that are only padded because then no information is lost. Then, we would be comparing the performance for adding black pixel as noise versus stretching an image to a specified size.

We then investigated the Imbalance Class problem by comparing performance results on imbalanced dataset and balanced datset of 64x64 re-scaled images. The accuracy results of each experiment were tabulated as shown below:

Table 2: Accuracy Comparison of None, SMOTE and ADASYN on 64x64 dataset

Epoch	None	SMOTE	ADASYN
5	87.12	86.87	88.77
10	89.10	90.25	90.71
20	90.52	93.00	94.10
50	88.37	89.67	91.25

The results show that balancing a data does indeed improve the overall performance, but since we only performed 1-fold, the results may not be an accurate representation. Despite balancing the dataset, the network is still incapable of differentiating dogs and cats with certainty. We can observe this by looking at the confusion matrices on Figure 3, Figure 16 and Figure 18. Looking at the middle matrix for each figure, we see that the ratio of misidentifying cats and dogs are approximately the same and this is because the up-sampling technique only duplicates our data and cannot reproduce new cat and dog images that hold distinctive traits.

When using an SPP the model was unable to perform as well as the above mentioned techniques, averaging 70-80% classification accuracy. The big benefit of using the SPP layer was no image resizing had to be done, however this also meant SMOTE and ADASYN could not be used as the libraries expect a uniform number of features per example. Not using these oversampling techniques could explain some of the loss in accuracy but also the size of the feature maps could have been an issue. Since many of raw images were rectangular in nature using square kernels could have prevented the model from correctly learning individual features. In addition the number of bins used in the SPP layer was quite small potentially using a higher number of bins could have increased accuracy. Another possible improvement, as mentioned in Kaiming et al [2], would be to resize images however instead of resizing all of the images to a static size, each image could be resize to one of many possible configurations depending on which configuration the image is closest to.

6 Conclusion

In conclusion, different resizing techniques can make a significant difference to classification accuracy, and balanced datasets do improve classification accuracy. From our experiments, scaling an image is generally a better approach as it retains the most information. Balanced datasets also improve the overall performance of the neural network. If an SPP layer were to be used then the architecture of the model would need to be rethought to gain a higher classification accuracy.

References

[1] Natural Images, Kaggle, August 2017
`https://www.kaggle.com/prasunroy/natural-images/`

[2] Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition, April 23 2015, Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun,
`https://arxiv.org/pdf/1406.4729.pdf`

[3] Handling imbalanced dataset in supervised learning using family of SMOTE algorithm, April 24 2017,
`https://www.datasciencecentral.com/profiles/blogs/handling-imbalanced-data-sets-in-supervised-learning-using-family`

[4] Visualizing Data using t-SNE, November 2018 Laurens van der Maaten, Geoffrey Hinton
`http://jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf`