**Ethnicity Classification using Neural Networks**

Christopher Audretsch

**Abstract**

Facial recognition and classification is a generally easy problem in the field of image processing. Various techniques are capable of producing highly accurate results. This paper will explore the use of convolutional neural networks to classify ethnicity based on an image of a face. Two neural net architectures are compared. The first is a generic architecture for image classification. The second is the specific architecture presented by Wang, He, and Zhao [3]. These networks are trained and tested on the UTKFaces dataset, which consists of ~24,000 photographs of people with labeled ethnicity white, black, Asian, Indian, or other [1]. Neither network preforms particularly well, with each having an accuracy of ~75%, far below the ~99% that is found in the paper of Wang et al. Reasons for this are discussed.

**1 Introduction**

Image classification is a popular task. In this paper, I will focus on the particular task of classifying ethnicity based on a photograph of a face. Image classification tasks like this tend to be well suited machine learning techniques, and it is generally not too difficult to achieve good results, especially in comparison to tasks in domains such as natural language processing.

Ethnicity classification has many practical uses. It could be used for providing demographic data for anything from restaurants to political rallies, as long as images of attendees’ faces are accessible. It could be a useful step in broader facial recognition tasks; it may be useful to identify ethnicities in order to determine familial relationship, for example.

From an academic standpoint, ethnicity classification is interesting because it is more difficult and complicated than many traditional image classification tasks such as animal classification or handwritten-number classification. The huge variety in facial features among those within a single ethnicity make this task nontrivial even for humans. In fact, it’s very possible that a good algorithmic ethnicity classifier would be better than nonskilled or even skilled humans at the task. Again, this differentiates it from some of the easier image classification tasks. On the other hand, this task is approachable enough that good results should be within the realm of possibility.

**2 Literature Review**

A relatively standard approach is exemplified well in Hosoi, Takikawa, and Kawade’s paper on facial ethnicity detection [2]. The idea here is to first identify facial features that are associated with different ethnicities. These features can be manually described, or they can come from a feature extraction algorithm (using other techniques from machine learning or image processing). Rather than apply a machine learning algorithm to the face as a whole, it is applied to the extracted features themselves. Hosoi’s paper uses support vector machines to do this part.

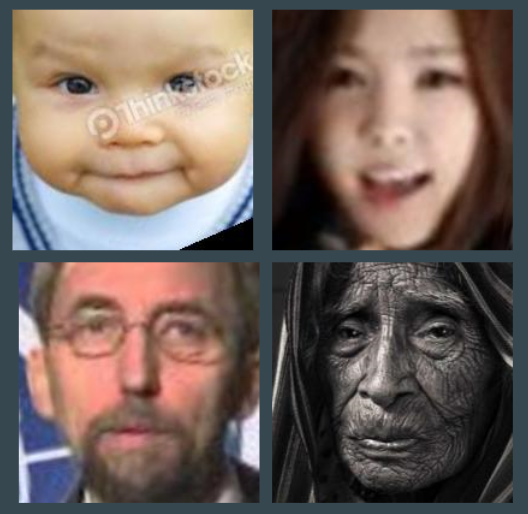
Another approach to this problem is to use a neural network. Wang, He, and Zhao present such a method [3]. They present a specific architecture for a convolutional neural network to use on these ethnicity classification problems. This architecture is used with various datasets with different ethnicities, including Black vs White, Chinese versus Non-Chinese, and Han vs Uyghur vs Non-Chinese. ty. The results of their paper are extremely promising. They report >=99.4% accuracy for each ethnicity on of these tasks. These accuracies are an improvement over other accuracies in the literature, which range from 94% to 99%.

This paper presents an attempt to replicate these positive results in a different context. A neural network approach will be used to classify five ethnicities – white, Asian, black, Indian, and other. Given there are five ethnicities rather than two or three, this task is harder and lower results are expected. Still, we expect this approach to be fairly successful to this problem. Furthermore, the specific architecture presented by Wang et all will be compared with a generic image-processing architecture.

**3 Dataset**

The UTKFaces dataset is used for this project [1]. This set consists of ~24,000 photographs of faces taken from various sources across the internet. Thus, there is a wide variety in lighting conditions, background, facial expressions, and so on. The dataset provides a processed set of these images in which the faces are aligned and cropped. This adds a consistency to the photographs which is helpful for algorithmic image processing, so this set of images is the one used for the problem.

Each face is classified as belonging to one of five ethnicities – white (42%), black (19%), Asian (14%), Indian (17%), or other (7%). Age and gender are also classified but not used. Ages range from 0 to 116, which exemplifies the variety in this dataset. A neural network trained on this dataset is learning how to classify ethnicity of women and men, old and young.



(4 images from the dataset. Note the variety in age, gender, lighting, expression, background, and blur.)

**4 Algorithm**

The overview for the algorithm is as follows:

1. Preprocess data.
2. Initialize neural net.
3. Train net on data.
4. Test net on data.
5. Repeat 3 and 4 until desired results are achieved.

This particular implantation was done in Python using the Pytorch library. Programming was done on a Google Collab notebook, which allowed for free use of Google’s cloud GPU to train the network.

**Data Preprocessing**

Preprocessing of data is essential for good results with neural networks. Images in the dataset are already aligned and cropped, but more work needs to be done. They are initially read as standard RGB images: 200x200x3 matrices with values ranging from 0 to 255. The following transformations are applied to each image before it is given to the network.

**RGB to Grayscale**

Despite the fact that we are dealing with a problem in which ‘skin color’ is relevant, what is more relevant is the relative brightness of features, rather than relative or absolute color. Variety in lighting conditions make color a lower source of predictive power than one might naively expect. The grayscale transformation is used in Wang’s paper and is used commonly in image preprocessing for neural networks. Besides having limited usefulness, the RGB values add another dimension to the data, bringing it from two to three dimensions. This requires the layers in the network to modified to accommodate the additional dimension, which adds time complexity.

**Image Resizing**

Every image in the dataset is conveniently the same size. However, 200x200 is not the ideal input size to a neural network. Computations might be able to be done more efficiently when the layers in the network have sizes equal to a power of 2. Furthermore, there is a tradeoff between resolution and time complexity; a 200x200 image is four times as large as a 100x100 image, but probably contains little more useful information. Images can be made smaller in order to speed up computation without losing much predictive power. Additionally, image resizing necessary when given an image of a different resolution. For the network based on Wang’s architecture, images are resized to 64x64. For the generic network, images are resized to 32x32. Resolutions lower than this are likely to lose significant amounts of valuable information.

**Conversion to Tensor**

Matrices are converted to a generalized data structure that the neural network implementation mandates. One important difference here is that image matrices contain values from 0 to 255, whereas tensors contain values from -1 to 1.

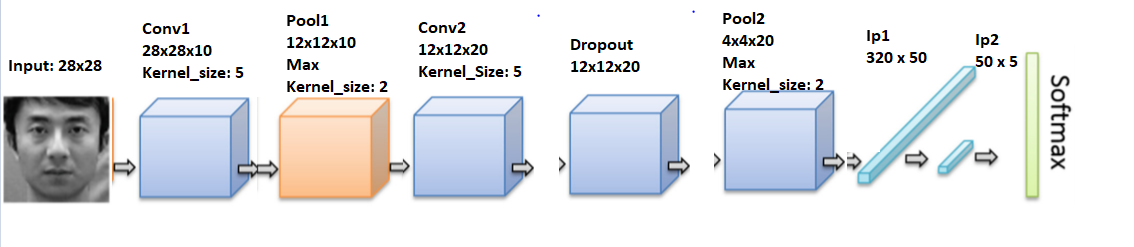
**Normalization**

It is the case that our neural network learns better if the training tensors are ‘normalized’ – that is, they are centered around 0 and their standard deviation is modified. After conversion to tensor, there is no guarantee that the It is also useful if the new ‘pixel’ values have mean zero (across all images). Thus, the value of *every* pixel in the image in the dataset is used to calculate a mean and standard deviation. Using these, the tensors are normalized.

**Neural Net Initialization**

A neural net can be abstractly conceived as an object consisting of different layers, each containing values that change as the network is trained. Constants such as learning rate, loss function, activation functions, and optimization function are also part of the network. As mentioned previously, two networks were compared: a network using a slightly modified version of Wang’s architecture, and a network using a generic image classification architecture. The generic architecture is based on a network used to classify handwritten digits, a popular toy problem for image classification with machine learning [4].

**Generic Architecture**

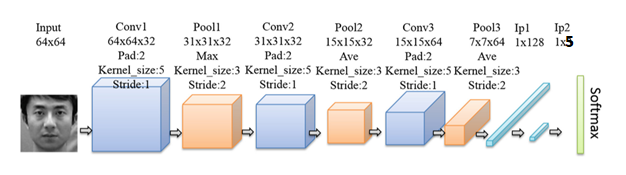
Here is a visual representation of the architecture:

The neural net consists of two convolutional layers. Convolutional layers work, as their name suggests, by convolving the input with some matrix (in this case, with kernel size 5). These layers have been shown to be essential parts of architectures for image processing. Max Pooling is a process to combine information from the multiple outputs given by the convolutional layer. The Dropout layer provides an effect in which some values in the net are randomly assigned to be zero. This can generally help with learning. The final two layers are fully connected layers. We can specify their dimensions in order to get the desired output dimension of 5, corresponding to the 5 ethnicities. Each value corresponds to a ’probability’ (non-normalized) that the image corresponds to that classification. Softmax is a function that normalizes these probabilities. Then, the network outputs the label with the highest probability as its classification.

The RElu function is used within the network as the activation function. Negative Log Likelihood is used as the loss function, and Stochastic Gradient Descent is used as the optimization technique. These are all standard choices for image classification tasks.

**Specific Architecture**

Here is a visual representation of the second architecture:



The broad idea is the same as before, using convolutional layers before fully connected layers. This network uses three convolutional layers rather than two and modifies the padding and stride parameters on the pooling and convolutional layers. Padding adds 0s to the edges of the input, and stride specifies how the layer ‘skips’ over some of the input. They also use a combination of max pooling and average pooling. The exact motivation for these particular choices is unknown to me, but it seems to have been successful for Wang et al. This network also uses RElu, Negative Log Likelihood, and Stochastic Gradient Descent.

**Training**

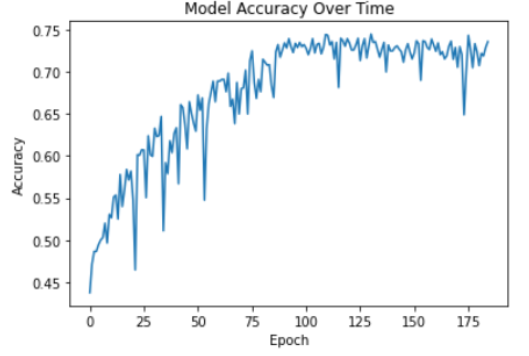
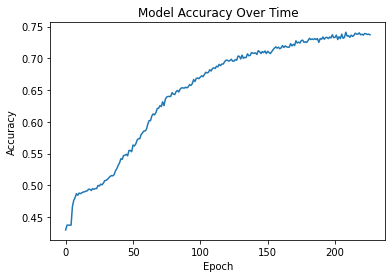
10,000 images were randomly selected from the dataset to be used as training data. 2,000 images were selected as testing data. A batch size of 512 was used with both architectures. This left ~12,000 images to be used as post-testing data. No differences in accuracy were found between testing and post-testing. This implies that overfitting is not a problem.

Training time was limited for these networks due to computational limits. The generic network was trained for 800 epochs, but the results were lost due to a user error. It was then trained from scratch for 250 epochs, using a learning rate of .01. The other network was trained for 200 epochs (requiring more computation time due to increased input dimension and number of layers). For the first 100 epochs it was trained with a learning rate of .01. This was modified to .005 and then to .003 for the next 100 epochs.

**5 Results**

The following graphs show the accuracy of the network over time.

Generic Architecture Specific Architecture

 (Specific Architecture Note: The occasional extreme dips seen in the graph are due to a bug in the code. They do not reflect the he accuracy of the network.)

When the generic architecture was ran for 800 epochs (not shown), it achieved an accuracy of ~78% by epoch 500 and did not improve for the next 300 epochs. While it’s possible that this could change with more testing, there is a good chance that 78% accuracy is an upper bound to accuracy here. Likewise, the specific architecture seemed to stop improving upon achieving 74% accuracy. However, it was given less time to train beyond this, so the implication that this is its maximum is weaker.

**6 Discussion**

Each network starts with an accuracy just above 42%, which is to be expected – 42% accuracy is trivially achieved by always classifying the ethnicity as ‘white’. Accuracy gain is rapid at the beginning and decreases over time, as is usual. Contrasts between the two networks can be seen in their corresponding graphs. The specific architecture’s accuracy generally varied much more from epoch to epoch. It initially learned faster than the generic architecture, though this leveled off. Surprisingly, the specific network did not do any better than the generic network after a sufficient amount of time had passed.

What could explain this? One possibility is that the specific architecture used in Wang’s paper was not really important. Another possibility is that our results are not really relevant; if we trained each network for another 1,000 epochs, perhaps the specific architecture would show better results. A third possibility is that Wang’s specific architecture was useful only for their problems – perhaps the data needs to be preprocessed in an additional way, or the architecture is only relevant to 2-ethnicity or 3-ethnicity problems.

Furthermore, the absolute accuracies of these networks -- ~75% -- are far lower than those seen in the literature (from both neural networks and other techniques). The obvious explanation is that classifying 5 ethnicities is harder than classifying 2 or 3. Beyond this, the specific dataset used could make a substantial difference in difficulty. The faces in the UTKFaces dataset vary in almost every way imaginable. This may not be the case in the datasets used in other papers. The more standardized the images – be it age, gender, expression, lighting condition, or something else – the easier the task will be. Another possibility for the lower accuracy could be a lack of training time (or training set size). These certainly have some effect. Especially given the variety in the faces, a 10x increase in the number of images could produce better results. Given all of this, it seems likely that the specific architecture for the neural network is not as important as one might think from the results in the Wang paper.

**7 Conclusion**

That the results found here are significantly worse than those found in the literature is to be expected. The combination of the problem being harder, the dataset being harder, the dataset being smaller, and the training size being limited explain this discrepancy. Further work could be done to see just how important each of these factors is. It would also be interesting to see how other ethnicity classification techniques would do given the same dataset and training time.

Still, the results were less promising than one would expect, given the incredible accuracies reported in the literature. Ultimately, though, 75% accuracy in this context is not trivial. It shows that meaningful learning is being done by the network. It’s possible that a 75% accuracy is as good as some humans would do on an equivalent task. This model might not be perfect, but it could be practically useful for some tasks in the real world.

**References**

1. UTKFaces. https://susanqq.github.io/UTKFace/

2. Wang, W., He, F., & Zhao, Q. (2016). *Facial Ethnicity Classification with Deep Convolutional Neural Networks. Lecture Notes in Computer Science, 176–185.* doi:10.1007/978-3-319-46654-5\_20

3. Hosoi, S., Takikawa, E., & Kawade, M. (n.d.). Ethnicity estimation with facial images. Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings. doi:10.1109/afgr.2004.1301530

4. Pytorch with the MNIST dataset. https://colab.research.google.com/github/rpi-techfundamentals/fall2018-materials/blob/master/10-deep-learning/04-pytorch-mnist.ipynb