Emotion Recognition from Facial Images using CNN and ResNet50 models

Abstract

This project explores the problem of limited data and imbalanced data in machine learning, specifically in the domain of facial recognition. Limited data is a challenge as it can impede the training process of machine learning models and result in overfitting and poor performance. Data augmentation, which is a technique where additional synthetic data are generated to increase the amount of training data, is one solution to this problem. This project investigates the use of data augmentation to create new and unique synthetic data and its impact on model performance. First, we briefly explain convolutional neural networks (CNNs), the history of data augmentation, and then an overview of the history of using generative adversarial networks (GANs) for facial emotional recognition tasks. Then we employ the use of two architectures of GANs, which are DCGAN and a pretrained stylistic GAN to create synthetic data. Finally, we compare the performance of two CNN architectures when synthetical data is included in the training set.

1. Introduction

During the Covid-19 pandemic, physicians were overwhelmed by the wave of new patients. Healthcare systems across the world were struggling to handle the magnitude of new, sick patients. To combat this, researchers sought to reduce the workload of physicians by employing a deep learning model (Oh, 2020). Their goal was to train a neural network to analyze chest X-ray images and classify whether or not a patient had Covid-19. However, because the pandemic was new and the situation was dire, the researchers were met with the problem of limited data.

Why would limited data be a problem for the researchers? Limited data poses a unique problem for machine learning. The performance of machine learning models is heavily dependent on the amount of data available for training. With limited data, models may not learn effectively and can result in poor performance. Meaning, if the data is limited, then the model training process may be impeded while trying to learn the contours and patterns of the data. Limited data is especially a problem for domains like disease diagnosis, object detection, and facial recognition.

Facial recognition using machine learning is a task that is highly complex and nuanced, especially given the various ethical concerns. To start, facial recognition is a difficult task as all faces generally look the same: two eyes, a mouth, a nose, etc. Because of this consistent pattern,

it can be difficult to delineate two faces from each other if the model is not specifically trained to spot granular differences. The model can also have poor performance on certain groups of people but great performance on other groups. This can happen if the training data is imbalanced with regard to race, age, or sex. Both of these concerns are present when we have a sizable amount of facial data. To that end, when we fold in a problem of limited data, the problems grow exponentially for a facial recognition model. Without a proper solution to the limited data problem, any hope of accurately recognizing faces should be promptly disposed of. One solution to this problem is data augmentation, which involves generating additional synthetic data to increase the amount of training data.

This project explores the use of data augmentation to create synthetic data. We also investigate the boundaries and limits of data augmentation when using synthetic data to retrain a model and the importance of sample size when creating synthetic data. It is essential to understand the impact of the inclusion of synthetic data on model performance and how sample size affects the goodness of the synthetically created images as synthetic data generation is posed to be a one-stop-shop solution to the limited data problem. For example, say that a researcher creates synthetic data from a limited dataset, trains a classification model with the synthetic data and the original data, then finds the model performance to be poor. Our project seeks to answer the following questions: Was it the model architecture's fault? Did the synthetic data generator have enough data? Was the synthetic data created well and does it resemble the data used to create it? Does the classification model have enough data to be trained adequately?

2. Background

Deep convolutional networks are powerful machine learning models that are commonly used in computer vision tasks. However, these models are prone to a common problem called overfitting, which is due to absence of big data. Inadequate training data leads to poor performance of the model (Alzubaidi et al., 2021) One solution to overfitting is transfer learning. Transfer learning is defined to be a technique where a model is first trained on a large dataset and then the model weights are then used for a different, but related task (Shin et al., 2016). Overfitting occurs when a model learns to fit the training data too well and doesn't capture the underlying patterns that generalize to new, unseen data. There are several solutions to the problem of overfitting and increasing the generalization performance (Wang and Klabjan, 2016). One solution is performing data augmentation.

Data augmentation is a set of techniques that can be used to increase the amount and variety of data in a training dataset, without collecting new data. There is a huge body of work performed in data augmentation for deep learning tasks as discussed in the survey (Shorten and Khoshgoftaar, 2019). The most common data augmentation techniques are image translations,

horizontal reflections, changing color channel configurations, cropping, image rotations, noise injections, kernel filters, mixing images, random erasing, and general adversarial net (GAN)-based augmentation (Shorten and Khoshgoftaar, 2019).

To restate, data augmentation is a method for tackling a problem of limited data. Data augmentation is defined to be a process that artificially enlarges a dataset using label-preserving transformations (Krizhevsky et al., 2012). Krizhevsky et al.'s paper offered new insight into data augmentation and how data augmentation can improve classification performance on difficult tasks. In their paper, Krizhevsky et al. retrain a classifier using the ImageNet data, as well as a synthetically inflated version of the ImageNet dataset using data augmentation. They use image translations, horizontal reflections, and reconfiguring the intensity of the RGB channels within the images. Using these techniques highly improved the model's performance.

Another common method of data augmentation is cropping. Simonyan and Zisserman use cropping, random horizontal flipping, and random RGB channel shift to augment their dataset (Simonyan and Zisserman, 2015). The goal of their paper was to vary the depth of a convolutional neural network and evaluate how performance of classifying images from the ImageNet dataset changes.

The next data augmentation that we will define and further investigate in this paper is GAN-based data augmentation. Goodfellow et al. first introduced GAN in their 2014 paper. This initial model included a generator and a discriminator. The generator takes images as inputs, and outputs new, unique images based off of the inputs. The discriminator then works to decide if the generated image is from the synthetic dataset or from the original, real dataset. (Goodefellow et al., 2014). Shortly after, Mirza and Osindero published their work, building off of Goodfellow, on conditional generative adversarial nets, where the generator network takes additional inputs, such as class labels, to control the generation process (Mirza and Osindero, 2014).

Continuing to improve GANs, Karras et al. proposed a new strategy for generating images, where the model is slowly expanded to increase the resolution of the generated images, which again improved the performance of classification tasks (Kerras et al., 2017). Further work includes that from Radford et al. in 2016, where they suggest a deep, convolutional GAN architecture (DCGAN) that had good performance in generating images using an unsupervised approach. (Radford, et al., 2016). This architecture is the one that we are using in this project. The DCGAN architecture was expanded by Odena et al., by explaining its use in the conditional setting, similar to the work by Mirza and Osindero, in which class labels are included in the training process (Odena et al., 2017). This paper is especially relevant to our project as we will be giving the input data with class labels.

A lot of work has been done in the space of facial emotion recognition using machine learning models and deep convolutional neural networks. Yichaun Tang developed a facial emotion recognition model using a Multiclass Support Vector Machine using the same dataset that we use and achieved the 'state-of-art' performance on the dataset (Tang, 2013). Following Tang, Sang et al. employed a CNN with fewer parameters than Tang's SVM on the same dataset and outperformed Tang (Sang et al., 2017). Using a different approach and dataset, Pranav et al., proposed using a two layer convolutional network to identify emotions stemming from five emotional categories and achieved an accuracy of 78.04% (Pranav et al., 2020). Their model, despite having a lower accuracy, is much simpler and less computationally expensive.

2.2 Definition of Limited Data and Synthetic Image Generation and work done

Data Augmentation has been used as the data space solution to the problem of limited and imbalanced data. Limited data is a problem as gathering enormous amounts of data and labeling it is a herculean task. In addition, effective classification with imbalanced data is an important area of research, as class imbalance, where the dataset is skewed with respect to the majority, can affect the performance of the model heavily (Johnson and Khoshgoftaar, 2019). Also, objects in realistic settings exhibit considerable variability, so to learn to recognize them it is necessary to use much larger training sets. There have been studies to solve limited data problems through various traditional transformations and synthetic image generation through GANs (Luis and Jason, 2017)

2.3 Brief History of Limited Data and data augmentation

Data Augmentation is a method for tackling a problem of limited data. Data Augmentation is defined to be a process that artificially enlarges a dataset using label-preserving transformations (Krizhevsky et al., 2012). Krizhevsky et al.'s groundbreaking paper offered new insight into data augmentation. In their paper, Krizhevsky et al. retrain the ImageNet model using two types of data augmentation.

2.5 Applications of Synthetic Image Generation with Facial Image Data

Data Augmentation refers to the data space solution of the problem of limited and imbalanced data. Limited data is a problem as gathering enormous amounts of data and labeling it is a herculean task. In addition, class imbalance, where the dataset is skewed w.r.t the majority, can affect the performance of our model heavily. Also, objects in realistic settings exhibit considerable variability, so to learn to recognize them it is necessary to use much larger training sets. Hence, data augmentation techniques like data warping and data oversampling are heavily

explored to solve these issues. These augmentations artificially inflate the training dataset size by either data warping or oversampling. Data warping augmentations transform existing images such that their label is preserved. This encompasses augmentations such as geometric and color transformations, random erasing, adversarial training, and neural style transfer. Oversampling augmentations create synthetic instances and add them to the training set. This includes mixing images, feature space augmentations, and generative adversarial networks (GANs). Hence, data oversampling can be very useful for problems with class imbalance.

3. Data

3.1 Data Source

We are using the FER2013, which was created by Pierre Luc Carrier and Aaron Courville as part of a larger, ongoing project (Goodfellow et al., 2013). To create the dataset, they used the Google image search API to collect images of faces that match a list of 184 emotion-related keywords such as “happy,” “enraged,” etc. Each of the 184 emotion-related keywords were mapped to seven emotions using the categories as used in the Toronto Face Database and these seven are what is used as the labels for the images (Susskind et al., 2013). The researchers combined the emotion-related keywords with demographic keywords related to race, age, and gender to hopefully return a dataset that would be balanced across each emotion and demographic profile.

After collecting the images, the researchers used OpenCV facial recognition to place boxes around each face in each image. Then, human labelers removed incorrectly labeled images, cropped the image if necessary, and filtered out some duplicates. Once the images were approved, they were resized to 48x48 pixels and converted to a greyscale. They set aside a subset of the images for a Kaggle Competition, which is the subset that we are using.

3.2 Data Features

In our data, we have 32,298 images across the seven categories. The training set contains 28,709 images and the test set contains 3,589 images. The categories are ‘happy,’ ‘neutral,’ ‘sad,’ ‘angry,’ ‘fear,’ ‘disgust,’ and ‘surprise.’ Below is a bar chart that visualizes the distribution of images across the emotions within our training set.

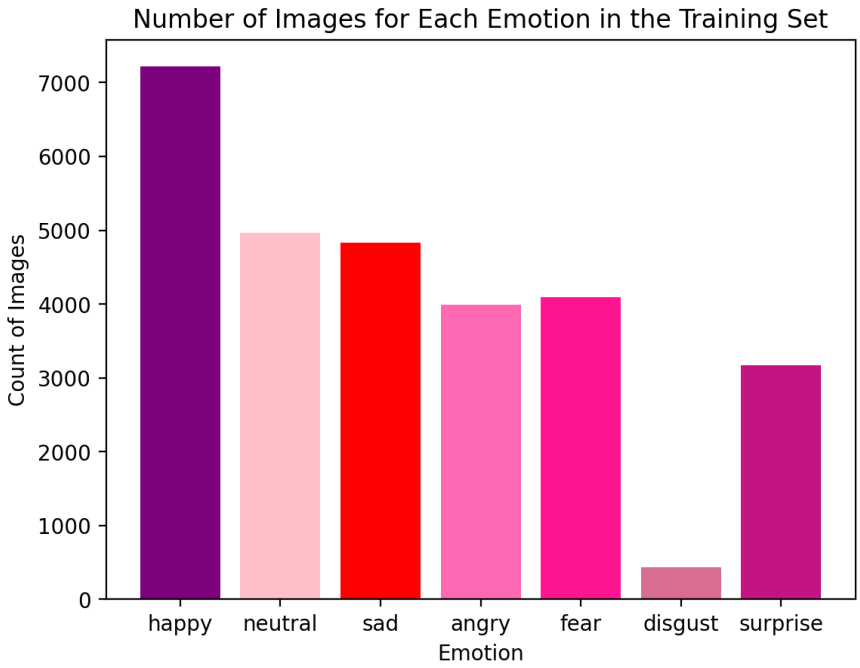


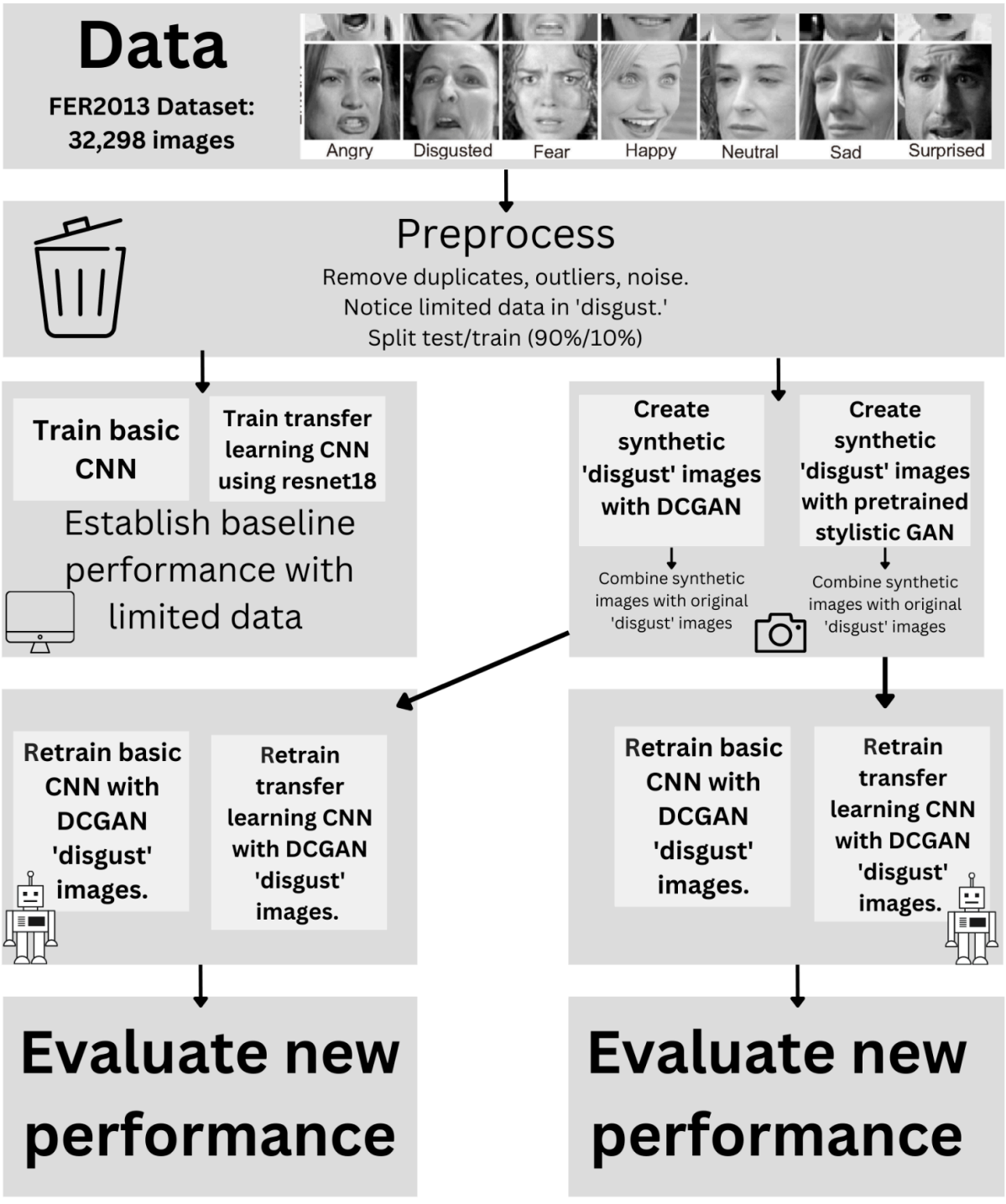
Figure 1: Bar chart of the number of images for each emotion.

It is important to note the imbalance of emotions. First, we note that we have an excess of images in the ‘happy’ category and, second, an insufficient amount of images in the ‘disgust’ category. If unchecked, this could cause us an issue when we begin training our classifier or when we use our synthetic image generator. When one category has far more records than the rest, the model may build that distribution into the classification scheme. In that, because it saw ‘happy’ more often than ‘disgust,’ then when faced with an image that could either be ‘happy’ or ‘disgust,’ choose ‘happy’. The same goes for a category that is highly misrepresented in the dataset. Because our ‘disgust’ category is limited to such a degree, our classifier or synthetic image generator may not have an adequate number of images in order to learn what ‘disgust’ looks like.

4. Experiments

In our experiments, we first preprocess our data. Then, we train two convolutional neural networks. One with transfer learning using resnet18 and one without transfer learning. The performance of these two models act as a baseline before we include our synthetic image data. From here, we generate 4,000 ‘disgust’ images using a pre trained stylistic GAN architecture and we generate another 4,000 ‘disgust’ images using the DCGAN architecture. Then, we retrain our two initial CNNs using the synthetically created images from the stylistic GAN combined with the original ‘disgust’ images and we retrain the two initial CNNs using the synthetically created images from the DCGAN combined with the original ‘disgust’ images. Our goal is to understand how using synthetically generated images using two different architectures respectively affect the

performance of two CNNs that each use a different architecture. On the next page is a diagram showing the workflow of the experiment.



4.1 Preprocessing

Just as in every other machine learning venture, we first preprocess our data. It is important to preprocess this data specifically because, as we explained earlier, the initial labels were derived from an API, then corrected by humans. The humans removed any duplicates that they saw and made the images grayscale and 48x48. Although we have full faith in the humans that oversaw the data creation and cleaning process, we chose to double check and ensure that the data is clean. If we were to have 'unclean' data, then our model could suffer from a myriad of issues ranging from bias from imbalanced data, overfitting from duplicates, and underfitting from noisy data and outliers. Because the goal of this project is to tackle the limited data problem by creating synthetic data, it is paramount that the data that we use as input for our GAN is as clean as possible. Otherwise, if our data is unclean, then our outputted, synthetically created images will be unclean as well. In turn, our classifier will suffer.

4.1.1 Duplicates

In this project, we define duplicates in the same way that we typically define duplicates. If two images are exactly the same, then they are duplicates. Below is an example of duplicate images with their file paths.

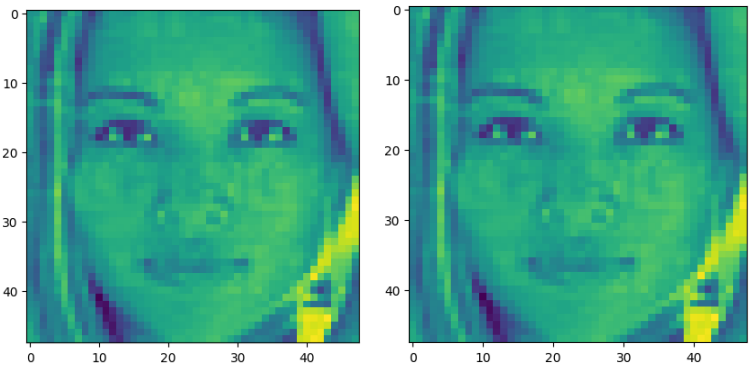


Fig 6:The image on the left has the name ''Training_87946315.jpg" and the image on the right has the name "Training_54214241.jpg"

To find duplicates numerically, we first hash each image with average hashing because it is computationally inexpensive and a relatively simple method to interpret, while still maintaining

performance. After each image is hashed, then we use the hamming distance with a threshold of zero. A hamming distance of zero indicates an exact replica or copy. Below is a distribution of the emotions that were called duplicates.

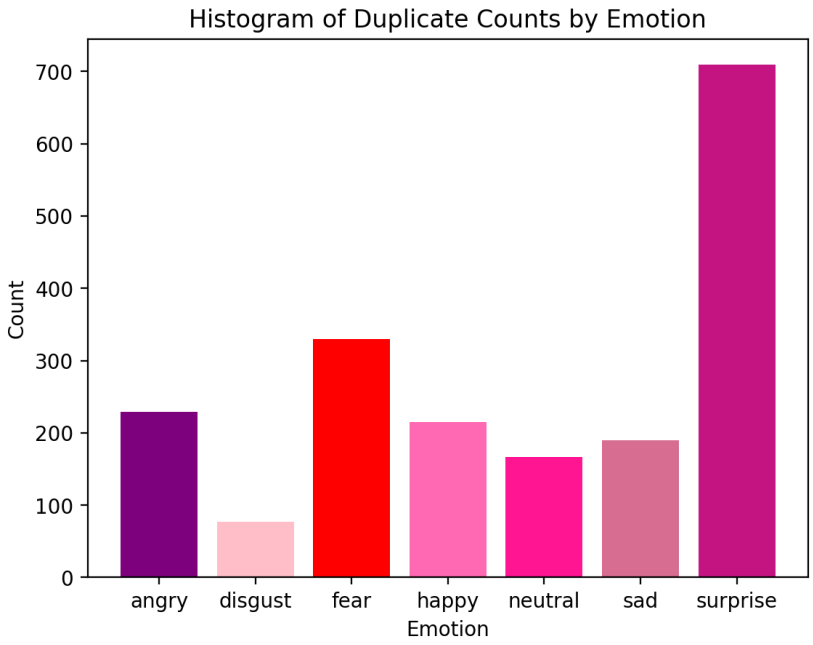


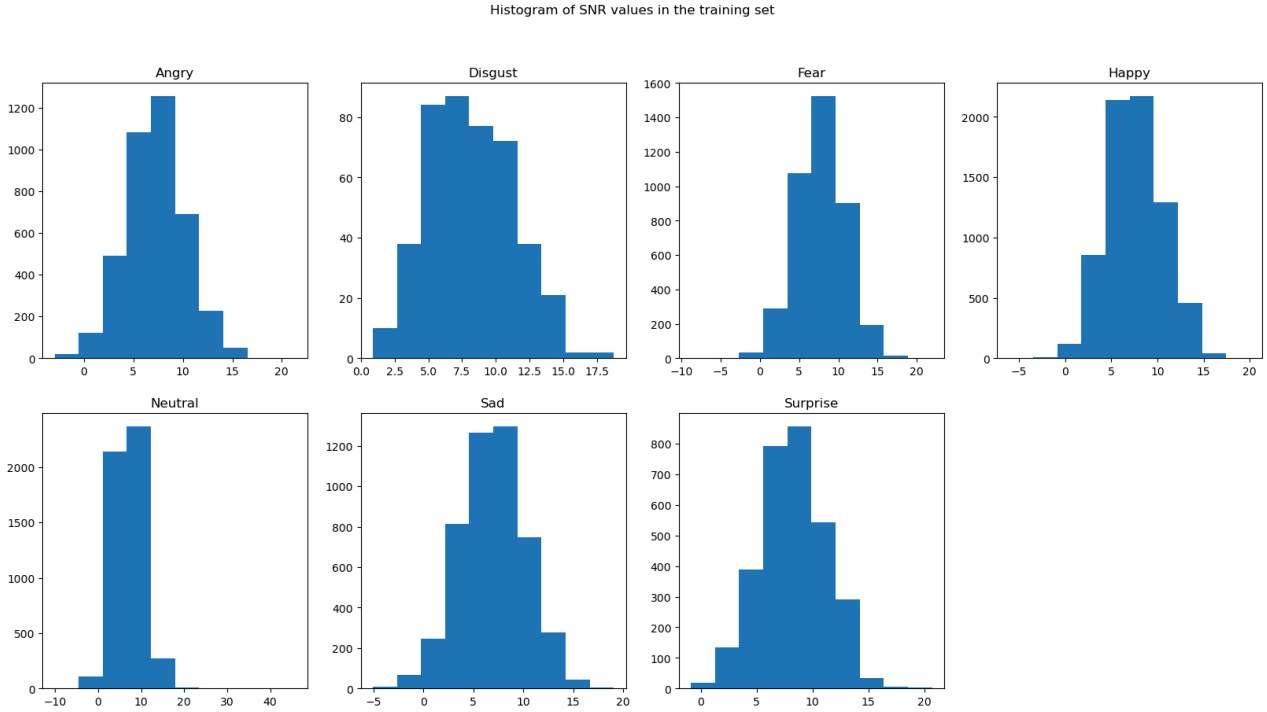
Fig 7: Distribution of duplicate images by emotion.

From this graph, we can see that 'surprise' has the greatest number of duplicate images, which is not in accordance with the overall distribution of images. It is important to delete duplicate images because duplicate images can inflate the distribution of images within each category and can create bias within the training set. For that reason, we delete 1917 images.

4.1.2 Noisy Data

Detecting noise in image classification requires a combination of visual inspection, image filtering, signal to noise ratio analysis, and feature extraction. After visually inspecting the images for noise, we applied signal to noise ratio (SNR) for every image, which is a metric that computes that ratio of the noise in an image relative to its signal. A low SNR reflects a high level of noise and vice versa for a high SNR. Based on that knowledge, we set a threshold for SNR, and any images below the threshold will be considered noisy. For the images which are noisy, we further evaluated the noisy images using histogram analyses where we used the standard deviation of pixel intensities as our measure for image noise calculation. If the standard deviation is above a certain threshold, we assume that the image has noise. This is because for images with low noise, the pixel values will be tightly clustered around the mean, resulting in a low standard deviation. For an image with high noise, the pixel values will be more spread out resulting in a higher standard deviation.

Below are histograms for each of our categories, denoting the sum of SNR within each category.

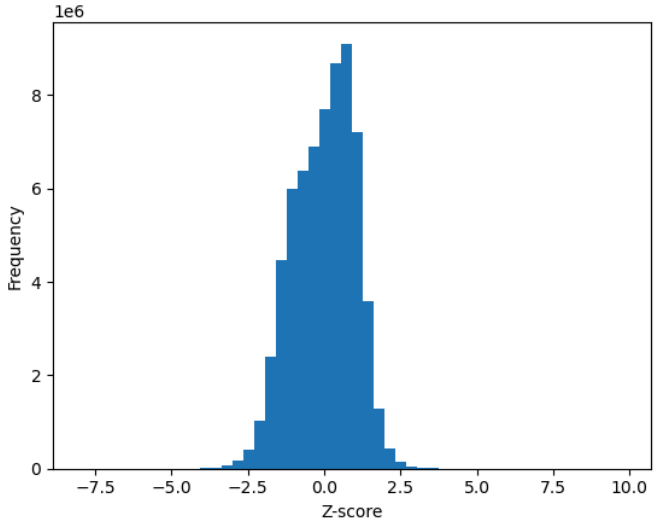


Once we detected our set of noisy images through these processes, we then applied fast fourier transformation (FFT) to the images to evaluate its frequency components and identify the high frequency components by zeroing out a small region around the center of the magnitude spectrum. We apply the inverse FFT to transform the filter frequency domain representation back into the spatial domain. We then extracted the high frequency components of the original image by subtracting the filtered image from the original gray-scale image. We applied a Gaussian filter to the high frequency components with a kernel size and then added the filtered high frequency components back to the filtered image to obtain the final result. Below is a histogram analysis of an example image, the magnitude spectrum for that image, and that image with a filter applied. .

Fig 3 : Distribution of Pixel intensities used for Histogram analysis, magnitude spectrum, and applied filter
(Img ref: /Training_4537951.jpg)

4.1.3 Outliers

In image processing, an outlier can be defined as a pixel value that is significantly different from the pixel values of the surrounding pixels or the expected pixel values of the image. This can happen due to various reasons such as noise, artifacts, or errors during image acquisition or processing. We have used the z-score algorithm for the purpose. The z-score algorithm is a statistical method used to detect outliers in a dataset, including image data. It identifies data points that are significantly different from the rest of the dataset by measuring how many standard deviations a data point is away from the mean of the dataset.In image processing, the z-score algorithm is commonly used to identify outlier pixel values. The algorithm is applied to each pixel of the image by first flattening the pixel values of the image into a one-dimensional array. Below is the z-score of the images in the training dataset.



The z-score threshold value for identifying outliers in image data depends on the specific application and the distribution of pixel values in the image. Generally, a z-score threshold value of three is commonly used as a starting point to identify outliers. The threshold value can be adjusted depending on the specific requirements of the image processing task and the desired trade-off between outlier detection and retaining useful data. In our analysis from the above visualization, we could have selected a z-score of three as threshold, however, this would have led to loss of a huge amount of image data. Hence, after careful deliberation we decided to go for a z-score with a value of five.

The algorithm was able to accurately identify non-facial images, images with occlusion, misclassified images, and images with unclear features. Some examples of the images detected as outlier as shown below:

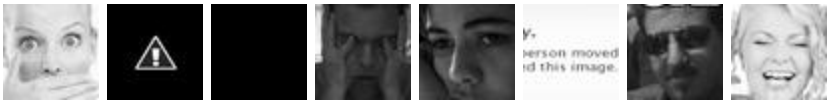


Fig 2: Examples of outlier images

The distribution of the images in the train data across the 7 categories after removing outliers is as shown below.

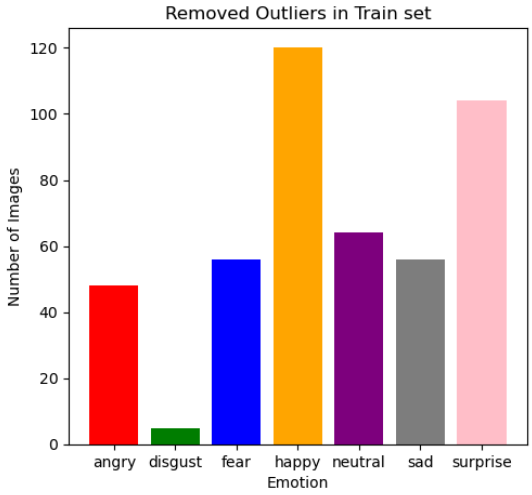


Fig 5: Distribution of images as outliers in train set

4.1.4 Color Discrepancies

In order to confirm that all images were greyscale, we checked the mode of each image individually. It is important that all images are of the same color model because if images had different modes, then it is possible that the classifier may infer a pattern based on the colors in the image instead of the nuances on each face. We found that all images were grayscale as reported in the initial paper.

4.2 Facial Emotion Classification

4.2.1 CNN Model:

- Evaluation Metrics:

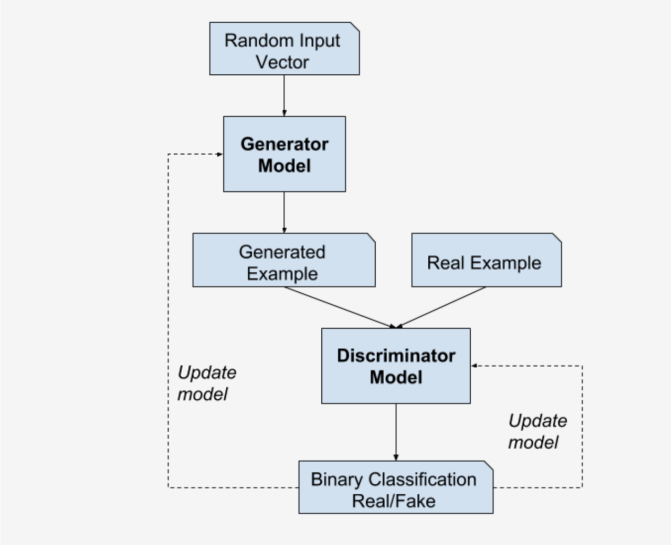
4.2.2 Transfer Learning:

- Model used
- Evaluation Metrics:

4.3 Synthetic Image Generator:

Data augmentation was performed by generating synthetic images belonging to the “disgust” emotion category using Generative Adversarial Network. GANs or Generative Adversarial Networks is a deep learning framework that is used to generate new data samples that are similar to the training data distribution. The GAN architecture comprises two models, namely the generator and discriminator. The

generator creates fake images that resemble the real training data, while the discriminator is trained to distinguish between real and fake images.



During training, the generator and discriminator models are trained and updated simultaneously in a competitive game-like manner. The generator tries to generate images that can deceive the discriminator, while the discriminator tries to identify the fake images produced by the generator from the real images in the training set. As the training progresses, the generator gets better at generating realistic fake images, while the discriminator becomes better at distinguishing between real and fake images.

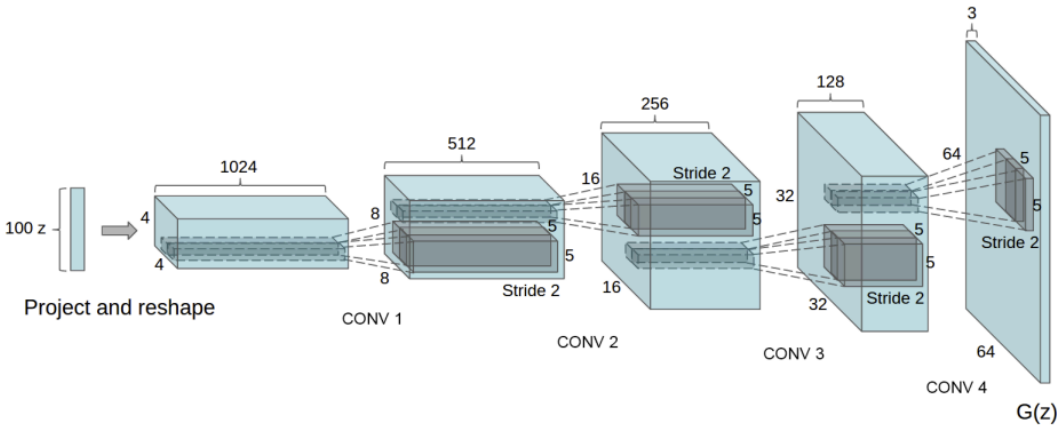
The ultimate goal of GAN training is to reach an equilibrium state where the generator produces images that are indistinguishable from real training data and the discriminator cannot distinguish between real and fake images with certainty. At this point, the generator has learned to capture the essential characteristics of the training data distribution, and it can generate new data samples that closely resemble the original data distribution.

We employed two different GANs for data augmentation dn compared the performance on both of them:

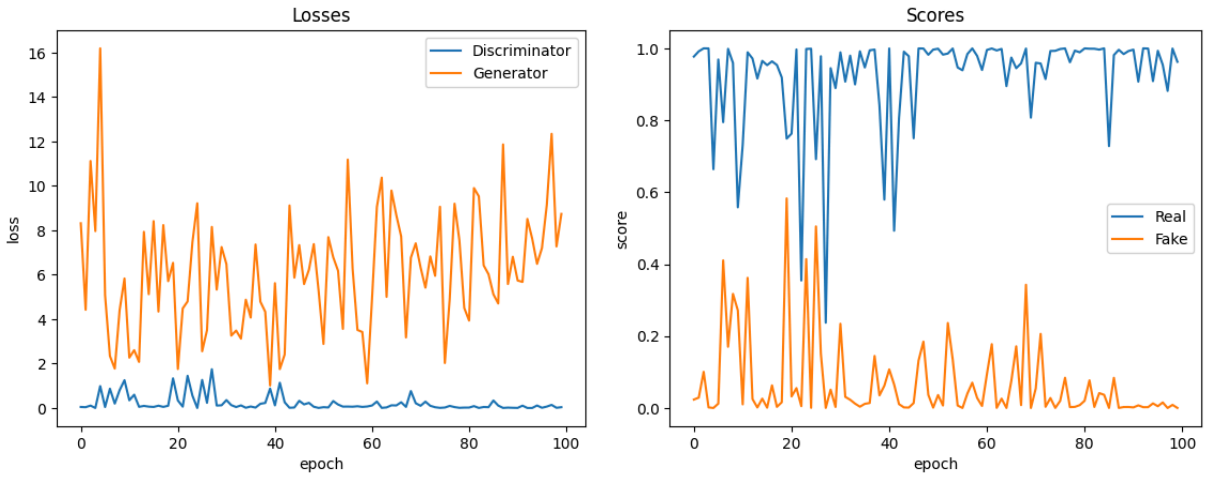
1. DCGAN
2. Pretrained Stylistic GAN

4.3.1 DCGAN

DCGAN or Deep Convolutional Generative Adversarial Network is an extension of the GAN framework, where the discriminator and generator networks use convolutional and convolutional-transpose layers, respectively. The architecture was first introduced in a paper on Unsupervised Representation Learning With Deep Convolutional Generative Adversarial Networks (Radford et al.,2016).



In a DCGAN, the discriminator is composed of strided convolutional layers, batch normalization layers, and LeakyReLU activations. It takes a 3x64x64 input image and produces a scalar probability indicating whether the input is from the real data distribution. On the other hand, the generator network comprises convolutional-transpose layers, batch normalization layers, and ReLU activations. It takes a latent vector, z, from a standard normal distribution as input and produces a 3x64x64 RGB image as output.
The strided convolution-transpose layers in the generator network help transform the latent vector into a volume with the same dimensions as an image. Additionally, the paper also provides recommendations on setting up the optimizers, calculating the loss functions, and initializing the model weights, which are crucial for successful DCGAN training.



4.3.2 Pretrained Stylistic GAN

5. Results

5.1 Retrain CNN on both Gan
- Evaluation metrics

5.2 Retrain Resnet on DCGAN
- Evaluation Metrics

53. Retrain Resnet on StyleGAn
- Evaluation Metrics

6. Conclusion

Works Cited

Alzubaidi, Laith, Jinshuai Bai, et al. “A Survey on Deep Learning Tools Dealing with Data Scarcity: Definitions, Challenges, Solutions, Tips, and Applications.” *Journal of Big Data*, vol. 10, no. 1, Apr. 2023, p. 46. *BioMed Central*, https://doi.org/10.1186/s40537-023-00727-2.

Alzubaidi, Laith, Jinglan Zhang, et al. “Review of Deep Learning: Concepts, CNN Architectures, Challenges, Applications, Future Directions.” *Journal of Big Data*, vol. 8, no. 1, Mar. 2021, p. 53. *DOI.org (Crossref)*, https://doi.org/10.1186/s40537-021-00444-8.

Goodfellow, Ian J., et al. “Challenges in Representation Learning: A Report on Three Machine Learning Contests.” *Neural Networks*, vol. 64, Apr. 2015, pp. 59–63. *DOI.org (Crossref)*, https://doi.org/10.1016/j.neunet.2014.09.005.

Johnson, Justin M., and Taghi M. Khoshgoftaar. “Survey on Deep Learning with Class Imbalance.” *Journal of Big Data*, vol. 6, no. 1, Dec. 2019, p. 27. *DOI.org (Crossref)*, https://doi.org/10.1186/s40537-019-0192-5.

Karras, Tero, et al. *Progressive Growing of GANs for Improved Quality, Stability, and Variation*. arXiv, 26 Feb. 2018. *arXiv.org*, https://doi.org/10.48550/arXiv.1710.10196.

Krizhevsky, Alex, et al. “ImageNet Classification with Deep Convolutional Neural Networks.” *Communications of the ACM*, vol. 60, no. 6, May 2017, pp. 84–90. *DOI.org (Crossref)*, https://doi.org/10.1145/3065386.

Mirza, Mehdi, and Simon Osindero. *Conditional Generative Adversarial Nets*. arXiv, 6 Nov. 2014. *arXiv.org*, https://doi.org/10.48550/arXiv.1411.1784.

Odena, Augustus, et al. *Conditional Image Synthesis With Auxiliary Classifier GANs*. arXiv, 20 July 2017. *arXiv.org*, https://doi.org/10.48550/arXiv.1610.09585.

Oh, Yujin, et al. “Deep Learning COVID-19 Features on CXR Using Limited Training Data Sets.” *IEEE Transactions on Medical Imaging*, vol. 39, no. 8, Aug. 2020, pp. 2688–700. *DOI.org (Crossref)*, https://doi.org/10.1109/TMI.2020.2993291.

Perez, Luis, and Jason Wang. *The Effectiveness of Data Augmentation in Image Classification Using Deep Learning*. arXiv, 13 Dec. 2017. *arXiv.org*, https://doi.org/10.48550/arXiv.1712.04621.

Pranav, E., et al. “Facial Emotion Recognition Using Deep Convolutional Neural Network.” *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2020, pp. 317–20. *IEEE Xplore*, https://doi.org/10.1109/ICACCS48705.2020.9074302.

Radford, Alec, et al. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. arXiv, 7 Jan. 2016. *arXiv.org*, https://doi.org/10.48550/arXiv.1511.06434.

Sang, Dinh Viet, et al. “Facial Expression Recognition Using Deep Convolutional Neural Networks.” *2017 9th International Conference on Knowledge and Systems Engineering (KSE)*, 2017, pp. 130–35. *IEEE Xplore*, https://doi.org/10.1109/KSE.2017.8119447.

Shin, Hoo-Chang, et al. “Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning.” *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, May 2016, pp. 1285–98. *IEEE Xplore*, https://doi.org/10.1109/TMI.2016.2528162.

Shorten, Connor, and Taghi M. Khoshgoftaar. “A Survey on Image Data Augmentation for Deep Learning.” *Journal of Big Data*, vol. 6, no. 1, July 2019, p. 60. *BioMed Central*, https://doi.org/10.1186/s40537-019-0197-0.

---. “A Survey on Image Data Augmentation for Deep Learning.” *Journal of Big Data*, vol. 6, no. 1, July 2019, p. 60. *BioMed Central*, https://doi.org/10.1186/s40537-019-0197-0.

---. “A Survey on Image Data Augmentation for Deep Learning.” *Journal of Big Data*, vol. 6, no. 1, Dec. 2019, p. 60. *DOI.org (Crossref)*, https://doi.org/10.1186/s40537-019-0197-0.

Susskind, J. M., Anderson, A. K., & Hinton, G. E. (2010). The toronto face database. *Department of Computer Science, University of Toronto, Toronto, ON, Canada, Tech. Rep*, *3*, 29.

Tang, Yichaun. *Deep Learning Using Support Vector Machines*.

Tang, Yichuan. *Deep Learning Using Linear Support Vector Machines*. arXiv, 21 Feb. 2015. *arXiv.org*, https://doi.org/10.48550/arXiv.1306.0239.

Wang, Xiang, et al. “A Survey on Face Data Augmentation.” *Neural Computing and Applications*, vol. 32, no. 19, Oct. 2020, pp. 15503–31. *arXiv.org*, https://doi.org/10.1007/s00521-020-04748-3.

Yang, Suorong, et al. *Image Data Augmentation for Deep Learning: A Survey*. arXiv, 18 Apr. 2022. *arXiv.org*, https://doi.org/10.48550/arXiv.2204.08610.