Facial Expression Recognition: Utilizing Digital Image Processing, Deep Learning, and High-Performance Computing

Francisco Reveriano*
Computer Engineering
University of Houston-Clear Lake
Houston, Texas
Reveriano@uhcl.edu

Unal Sakoglu
Computer Engineering
University of Houston-Clear Lake
Houston, Texas
Sakoglu@uhcl.edu

Jiang Lu
Computer Engineering
University of Houston-Clear Lake
Houston, Texas
Luj@uhcl.edu

ABSTRACT

The purpose of this project was to analyze which image pre-processing technique was most beneficial in improving the performance of Facial Expression Recognition through Deep Learning and High-Performance Computing. Contrary to our expectations, the results obtained in this work showed that deep learning does not significantly benefit from various commonly used image pre-processing techniques such as resizing, smoothing, or edge detection. The results confirm previous findings that an increase in accuracy is obtained by increasing the size of the training dataset. This study proceeds to show that the increase in training data size can easily be handled by the High-Performance Computing (HPC) cluster provided by the Pittsburgh Supercomputing Center (PSC) through XSEDE.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; *Supervised learning by classification*; Batch learning.

KEYWORDS

Facial Expression Recognition, Deep Learning, Neural Network, High-Performance Computing, Digital Image Processing, Big Data, GPU, XSEDE

ACM Reference Format:

Francisco Reveriano, Unal Sakoglu, and Jiang Lu. 2019. Facial Expression Recognition: Utilizing Digital Image Processing, Deep Learning, and High-Performance Computing. In *Practice and Experience in Advanced Research Computing (PEARC '19)*, July 28-August 1, 2019, Chicago, IL, USA. ACM, New York, NY, USA, 7 pages. <https://doi.org/10.1145/3332186.3333039>

1 INTRODUCTION

Facial expression recognition (FER) is one of the most critical non-verbal channels through which robots can recognize a human's internal emotions. Facial Expressions are broken down into six basic emotions: anger, disgust, fear, happiness, sadness, and surprise.

*Main author and contributor to this research.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

PEARC '19, July 28-August 1, 2019, Chicago, IL, USA

© 2019 Association for Computing Machinery.

ACM ISBN 978-1-4503-7227-5/19/07...\$15.00

<https://doi.org/10.1145/3332186.3333039>

Most facial recognition algorithms have been based on some type of neural network to attempt effective classification in a supervised manner. Unfortunately, most attempts have proven less than adequate. The prevalence of powerful multi-core and many-cored based architectures in modern HPC systems have accelerated the training process of deep neural networks. Unfortunately, these resources are very limited and every attempt to improve accuracy should be studied.[2] [6] [13]

Most attempts in utilizing Deep Learning for FER have focused solely on the training or algorithmic aspect of the data. Very few studies have taken a comprehensive approach to analyze the effects of image pre-processing on FER and Deep Learning.[6] [11]

To accomplish this objective, we worked with subsets of three different facial expression databases. The project was divided into two main sections. The first section involved analyzing and testing several image processing techniques in the color and grayscale domain by utilizing MATLAB Resnet50. The second part of the project involved utilizing the best image processing technique from the first section to pre-process the Kaggle Facial Expression Database and then utilize deep learning to train this new database. The training was performed by utilizing XSEDE resources, specifically the Bridges GPU supercomputer maintained by the Pittsburgh Supercomputer Center (PSC).

The results in our first section highlight the importance of testing techniques on data subsets before proceeding to use more expensive computing resources. The image processing results showed that various image pre-processing steps negatively affects the accuracy of deep learning. The second section showed that the ability to train an extremely large dataset through multiple epochs significantly increased the accuracy of our model.

2 BACKGROUND

In this section, the datasets, image processing techniques, and neural networks utilized in this study will be discussed. Emphasis is placed on the organization of each dataset.

2.1 Datasets

This study utilized the subsets of three different datasets. The dataset utilized for all color image processing was the MMI Facial Expression Database. Two other datasets used for the grayscale image processing were the Cohn-Kanade Dataset and the Kaggle Facial Expression (2013) dataset. All of these datasets are described in detail in the following subsection.

Each dataset exhibited an uneven distribution in images for each emotion category. This required the need to balance each dataset for effective training. The decision to balance the datasets was

necessary to make sure that the results for each emotion were even. That way we could conduct the study under a more controlled situation.

The images chosen in each balanced dataset were randomly chosen from the full available images in category. This was done each time the model was run to increase variation.

2.1.1 MMI Facial Expression Database.

The MMI Facial Expression database is a color database that consists of over 2900 videos and high resolution still images of 75 subjects. [10] [16] The advantage of using this database is that each frame is fully annotated with the emotion label and FACs Action Unit (AU). This is one of the best databases available for Facial Expression but unfortunately there were several disadvantages.

The database has over 2900 videos and images but only a small subset is classified as single emotion. A secondary problem was that each recording shows the actor performing the emotion in a frontal and posterior position at the same time. This meant that in order to use this database we were required to: organize by emotion label, convert video to frame, and then crop the frame to just show only the frontal image. The result of this work is visible in Figure 1.

Figure 1: Display a random selection of the MMI Database with no image cropping.

The process eventually led to a dataset containing only 876 images. The training required the balancing of the database to 110 images for each emotion label which meant 660 images total. The distribution is shown in Table 1.

As previously described, the dataset was made from videos of actors performing the desired emotion. The process of breaking down each video into frames resulted in a very ununiform dataset furthered visible in Figure 1.

MMI Database		
Emotion	Original	Balanced
Anger	149	110
Disgust	165	110
Fear	110	110
Happiness	158	110
Sadness	155	110
Surprise	139	110
Total	876	660

Table 1: MMI Database: displays the total in each category and the balanced total for each emotion category.

2.1.2 Cohn-Kanade Database.

The second database utilized was the Cohn-Kanade Database. [7] [8] [12] This database was utilized to analyze the performance of original grayscale images in Deep Learning. The advantage of this database was its clear emotional annotations. Furthermore, the database was originally in the image form with each subject expressing the emotion in a frontal position. This meant that there was no conversion or cropping required. An image of each emotional category in the Cohn-Kanade database is shown in Figure 2.

Figure 2: Display a random selection of the Cohn-Kanade dataset.

Due to time constraints, I was, unfortunately, unable to organize the full database. The number of images utilized for this database totaled 1549 images. The training required the balancing of the database to 104 in each category. Table 2 outlines the final distribution.

Cohn-Kanade Database		
Emotion	Original	Balanced
Anger	310	104
Disgust	276	104
Fear	104	104
Happiness	398	104
Sadness	110	104
Surprise	351	104
Total	1549	624

Table 2: Cohn-Kanade Database: displays the total in each category and the balanced total for each emotion category.

The Cohn-Kanade was one of the first pioneering facial expression databases created. This meant that it had the disadvantage of having lower picture quality. This disadvantage made it an ideal case to test image pre-processing on. The database was also in the .jpg file which made it easier to incorporate to the workflow.

2.1.3 Kaggle Facial Expression Database.

The third and most important database utilized was the Kaggle Facial Expression Database. [4] This database main advantage is its size. The Kaggle database consists of 28,709 samples in the training set and 3,589 examples for the test set. Figure 3 displays an image sample of each emotion category in the dataset.

Due to computing limitations, only a small subset of the Database was used in the Image Pre-Processing tested on MATLAB. The total number of images used on the MATLAB testing totaled 19675. The requirement to balance the training set meant that the final total was 4266 images. This uneven distribution shows the

Figure 3: Display a random selection of the Kaggle dataset.

Kaggle Database		
Emotion	Original	Balanced
Anger	4305	711
Disgust	711	711
Fear	4201	711
Happiness	4803	711
Sadness	2642	711
Surprise	3013	711
Total	19675	4266

Table 3: Kaggle Database: displays the total in each category and the balanced total for each emotion category.

difficulty of making a large and stable dataset for Deep Learning. The distribution is shown in Table 3.

The disadvantage of this database is that each image is a 48x48 grayscale images of faces. A further disadvantage in the database is the lack of quality control which means several unnecessary images. This two disadvantages are the main reason we decided to see if preliminary digital image processing could improve our results.

2.1.4 Kaggle + Cohn-Kanade Database.

Since the Kaggle FER database was the main database that was going to be utilized to test our model in HPC we proceeded to combine the Cohn-Kanade and the subset of the Kaggle FER Database. This was deemed appropriate for testing because both databases were originally grayscale. This new Kaggle + Cohn-Kanade Database distributed to 815 images per category. It was hoped that this larger size would also show whether it made sense to combine datasets for future work.

2.1.5 Full Kaggle Database.

The full Kaggle FER database was used only for the High-Performance Computing section of the study. We were unfortunately unable to test the whole dataset on MATLAB Resnet 50 due to computing limitations.

2.2 Digital Image Processing Techniques

The purpose of this study was to explore the effect of Digital Image Processing on Deep Learning for Facial Expression Recognition. The combination that yielded the best results would then be used on the training of the model in a High-Performance Computing Environment. This meant that image processing techniques tested were focused on seeing if the different resizing options, filtering, or edge detection improved the performance of the models.

2.2.1 Color Image Processing.

For the color dataset testing we utilized the Viola-Jones Cropping, Nearest Neighbor Interpolation, Bicubic Interpolation, Bilinear Interpolation, Gaussian filtering, and Canny Edge Detection. Each technique will be discussed briefly.

2.2.2 Grayscale Image Processing.

The three image processing techniques utilized on the grayscale datasets were: Bicubic Interpolation, Bilinear Interpolation, Nearest Neighbor Interpolation, Gaussian Filtering, and Canny Edge Detection. A brief comment of the purpose of the technique will be described in the following sections.

2.2.3 Viola-Jones.

The Viola-Jones algorithm is a popular Haar-Cascade model that is used for face detection. [17] The algorithm is designed to produce a box around the object if a face is detected. We proceeded to alter the algorithm to crop the image to that only in the target box. This was an attempt to see if the background noise could be reduced.

2.2.4 Resizing.

The use of Resnet 50 on MATLAB required the resizing of the images to a 224 x 224 size. To test to see if the resizing processing method had any effect on accuracy, We proceeded to utilize the bicubic, bilinear, and nearest neighbor resizing algorithms.[3] An important note is that MATLAB utilizes the Bicubic Interpolation as default.

2.2.5 Gaussian Filtering.

The Gaussian Filter is meant to smooth and sharpen the image. Although, we tested the Gaussian Filter on both the color and grayscale database it was primarily meant for the Kaggle database. As mentioned previously the images were a size of 40 x 40 and of very poor quality. The results are visible in Figure 4.

Figure 4: Display the effects of the Guassian Filter on Kaggle image.

The filter was meant to improve the image quality and hopefully to make the image information clearer for the model.

2.2.6 Canny Edge Detection.

Edge detection is one of the major components for classical Facial Detection and Face Recognition. This combined with the fact that most studies use the Facial Action Units to classify emotion made a relevant case for testing. [14] The result of the Canny Edge detection is visible in Figure 5.

Figure 5: Display the effects of the Canny Edge Detection.

The Canny Edge detection was chosen because it generally provides better results than the Prewitt Edge Detection Algorithm.

2.3 Network Architecture

The two parts of this study were the image processing testing and the training of the database on the High-Performance Computing Platform. This required the use of two different networks for testing.

2.3.1 Resnet 50.

The image processing testing was done utilizing MATLAB Resnet50. MATLAB was chosen due to its programming ease for Digital Image processing. In fact, many of the Image Processing Algorithms are already available in the Image Processing Toolbox. Resnet50 is a convolutional neural network that was trained on more than a million images from the ImageNet Database and is 50 layers deep.[1][5] The flexibility provided by the model made it an ideal candidate for the utilization of multiple broad testing.

2.3.2 Neural-Network for HPC.

The second part of the study required the training of a neural network on a High-Performance Computing Environment. This required designing a neural network utilizing Tensorflow and Keras on Python. Furthermore, to take advantage of the High-Performance Environment required every attempt to parallelize the model and allow for GPU training.

The network designed has ten main layers divided into two sections. The first section includes 5 convolutional layers, 1 max-pooling layer, and 2 average pooling layers. The first convolutional layer has 64 5x5 filters. The second and third convolutional layer each have 64 3x3 filters. The fourth and fifth convolutional layers have 128 3x3 layers. The second section of our CNN contains 2 fully connected layers, each containing 1024 neurons.

2.4 XSEDE HPC

The main goal of the study was to create and train a neural network in a High-Performance Computer environment utilizing the resources of XSEDE.[15] The image processing testing and training was used on a regular desktop computer under the impression that the results would lead to image pre-processing being combined with the constructed neural network and trained in the HPC environment.

2.4.1 Bridges GPU.

High-Performance Computing (HPC) is defined as the use of multi-core and many core systems to solve computational problems that are not feasible on regular computers. The main resource made available by XSEDE was the Bridges GPU supercomputer administrated by the Pittsburg Supercomputer Center (PSC). [9]

The Bridges GPU resource consist of 48 nodes. Sixteen of those nodes are NVIDIA Tesla K80 GPU Nodes. These nodes each hold two GPU and two Xeon CPUs. These nodes are idealized for parallel computing such as deep learning. The other 32 nodes are NVIDIA Tesla P100 GPU nodes with each containing one very powerful GPU.

The environment initialized in the Bridges GPU was Tensorflow Version 1.8 and Python 3 specific for GPU computations. This environment proceeded to initialize the Cuda 9.0 Furthermore, OpenHPC was also initialized to allow the appropriate pre-packaged binary components to initialize. This initialization allowed for effective parallelization of the code and faster processing time.

3 RESULTS

The results are divided into three main sections. The first discuss the results of testing the color image pre-processing. The second discusses the results of testing the grayscale image pre-processing. The third discusses the training and testing of the Neural Network on the HPC.

In order to achieve accurate and comparative results the same model structure was used for each of testing of an image processing technique. The datasets were divided to use 75 percent of the images for training and the other 25 percent for testing. This meant that the balanced set utilized in the testing/training of each model consisted of a random selection of images. Thus, the accuracy results of each time the model was run would naturally vary. This was solved by running each model ten times and using the average mean to analyze each image processing technique.

3.1 Color Image Pre-Processing

The Color Image Processing displayed surprising results. It was originally expected that image pre-processing would improve the accuracy of the deep learning network. The MMI dataset was first tested without any image processing performed other than the Bicubic Interpolation to resize to the necessary size for Resnet50. The second test involved further cropping the images by using the Viola-Jones Algorithm and then a Bicubic interpolation for resizing. The third test attempted used the Viola-Jones Algorithm for cropping and the Nearest-Neighbor Interpolation for resizing. The fourth test performed consisted of the Viola-Jones Algorithm for cropping and the Bilinear Interpolation for resizing. The fifth test further enhanced the image processing pipeline by using the Viola Jones Algorithm to first crop the image, the Bicubic interpolation for resizing, and then a Gaussian filter to smooth the image. The final and sixth test involved using the Canny Edge detection algorithm on the original non-processed dataset. Figure 6 shows that image processing failed to improve the accuracy of the results.

Summary of Color Results		
Algorithm #	Image Processing	Average Accuracy
1	No Image Processing	0.9704
2	Viola-Jones Algorithm	0.9117
3	Viola-Jones + Nearest Neighbor Interpolation	0.8790
4	Viola-Jones + Bilinear Interpolation	0.8704
5	Viola-Jones + Bicubic Interpolation + Gaussian	0.9339
6	Canny Edge Detection	0.8198

Figure 6: Summary of Color Image Pre-Processing results.

The Viola-Jones was meant to reduce the image noise fed to the model, but it apparently confused the model by reducing the image content. This study showed that the Viola-Jones algorithm was not effective in an image that clearly showed the face. This does not mean that the algorithm should not be used in images with multiple faces or objects.

The comparison of the two different interpolation techniques showed that the resizing technique utilized was insignificant. The Gaussian Filter mixed with the Viola-Jones and Bicubic-Interpolation was the most successful technique tested but it still had a 6.61 percent error when compared to the non-processed database. The Canny-Edge detection was the worst performing image processing technique with an 18 percent error when compared to the original database.

The probable reason that image pre-processing did not help the dataset is due to a faulty dataset. The MMI Database is a helpful resource that had several disadvantages previously described. The uniformity of the dataset might mean that when we altered the dataset through image pre-processing we decreased this uniformity.

3.2 Grayscale Image Pre-Processing

The testing of the Grayscale Image Pre-processing was imperative to decide which pre-processing technique to use for the ultimate training of the full Kaggle Dataset on the HPC environment. Furthermore, the Grayscale Image pre-processing was performed to compare and validate the results of the Color Image Pre-processing. This would avoid reaching any wrong preliminary conclusions.

Five different testing/training runs were performed on the dataset. We first tested trained the Kaggle database with only the default Bicubic Interpolation for resizing as a control variable. I then proceeded to test the combined Kaggle-Cohn-Kanade database. This database utilized only the default Bicubic Interpolation for resizing. This combined database was created to test if a modest increase in database size would improve results. The third test utilized the combined Kaggle-Cohn-Kanade database to test the effects of the Nearest Neighbor Interpolation for resizing on the accuracy. The fourth test continued to utilize this combined database and tested

the effects of the Gaussian filter on the original combined Database that was first resized by the default Bicubic Interpolation. The fifth and final test used the combination of the Bicubic Interpolation for resizing, followed by the Gaussian filter for smoothing and ended with the Canny Edge Detection. The result of the image pre-processing is visible in Figure 7.

Summary of Grayscale Results			
Algorithm #	Database	Image Processing	Average Accuracy
1	Kaggle	No Image Processing	0.3829
2	Combined	No Image Processing	0.3976
3	Combined	Nearest Neighbor Interpolation	0.3388
4	Combined	Bicubic Interpolation + Gaussian Filter	0.3565
5	Combined	Bicubic Interpolation + Gaussian Filter + Canny Edge Detection	0.3500

Figure 7: Summary of Grayscale Image Pre-Processing results.

The trend visible in the color image pre-processing continued to be visible in the grayscale preprocessing. The results, in Figure 7, clearly show that the Nearest Neighbor Interpolation for resizing was the worst performing with an average change of almost 12 percent in classification accuracy. Interestingly, the Gaussian Filter and Canny Edge Detection resulted in similar results. The most important conclusion from this section is that a modest increase of 104 additional training image available significantly improved the classification accuracy of the model by almost 4 percent. This meant that the accuracy could easily be increased by simply increasing the size of the training set.

3.3 XSEDE HPC

Originally the goal of this study was to integrate the best image pre-processing technique with the deep learning neural network we designed in the hope to get more accurate results. The image pre-processing testing we conducted showed that image pre-processing did not improve deep-learning based classification accuracy in emotion classification from facial images. In fact, it showed that it would reduce the accuracy of the deep learning model. The image testing in the previous sections did show that a modest increase of data would significantly improve the performance of our network.

This meant that it was not unreasonable to expect better results when we processed the whole Kaggle Dataset. Of course, processing a dataset this sized posed its own series of challenges.

To test the deep learning model, we first ran it on the regular computer environment used for all the image pre-testing. The results, visible in Table 4, show the difficulties of conventionally working with big data.

The training of our neural network model utilizing 10 epochs took our conventional computer (Dell Laptop i7 2.220 GHz 16 GB of RAM and NVIDIA GTX1060) a total of 1765 seconds (approximately 30 minutes). As Table 6 shows, during this time period, we were

Regular Computer Environment			
Epoch #	Runtime (s)	Loss	Accuracy
1	181	1.7948	0.2603
2	175	1.5765	0.4023
3	176	1.3822	0.4646
4	173	1.2645	0.5142
5	174	1.1830	0.5503
6	177	1.1275	0.5728
7	172	1.0654	0.5973
8	173	1.0114	0.6177
9	185	0.9560	0.6406
10	179	0.9071	0.6607

Table 4: Conventional Computer Environment Results

barely able to reach 0.51 accuracy in the training set. This accuracy is the accuracy of the model on the data which the model was trained. The loss represents the fraction of instances on which our predictions disagree with the ground truth. Generally, a lower loss value is better. Table 4 clearly created the requirement and need for a much more powerful computing environment.

The training of our neural network in the HPC Bridges GPU environment was very successful. The network was able to run each epoch in 5 seconds. So, we were able to reach 50 epochs in 250 seconds (4.16 minutes). This was an extremely impressive time because to reach 50 epochs it would take our average computer 14,750 seconds (246 minutes). This truly showed the power of the Bridges GPU. The results are summarized in Figure 8.

Figure 8: Summary of Running the model in HPC.

More importantly we were able to quickly reach and analyze the saturation rate of the model in 30 epochs. The accuracy and loss function of the training are almost exact to the accuracy and results visible when we ran the model in the conventional computer environment. This was expected since it was the same model used in both environments.

The testing of the model on the test dataset which contained a total of 3,589 images. At 100 epochs the model was able to deliver 0.57 in classification accuracy. This is visible in Figure 9.

This accuracy confirmed the results of our test image pre-processing tests in grayscale. It confirmed that the image pre-processing techniques utilized in our testing were unnecessary. Crucially, it showed

Figure 9: Result of testing model in HPC.

that an addition of images to the training dataset significantly improves the performance of the model.

4 CONCLUSION

This study showed that various commonly used image processing techniques such as resizing, smoothing, and edge detection did not improve the classification results of deep learning for emotion classification using various face image datasets. We can conclude that, in solving the problem of emotion recognition from images, computing resources can be better allocated to improving the parameters of deep learning networks rather than allocating them to commonly used image processing of the images. If the goal is to improve the accuracy of a model, resources could also be spent on improving the size of the training dataset instead. The most significant possible improvement is the increase of the size of the dataset. We found that the most impactful improvement was the increasing of the size of the dataset.

This study also showed that a High-Performance Computing Environment allows for the quick processing of an extremely large dataset. This gives the researcher the unique opportunity of not having to worry about the processing power required to train such a large dataset.

Deep learning is a powerful technique which can learn discriminatory features from data. One downside of this is that the features that the deep learning network uses are not easily interpretable. These mimic the enormous complexity of how the human brain encodes various memories and tasks in the connections/synapses of the neurons, as the deep learning is modeled on. Deep learning combined with HPC is truly revolutionizing the machine learning, artificial intelligence, classification, and if understood fully, it may even help us understand how the human brain functions. This would eventually succeed in helping us create artificial intelligence that can recognize and understand our complex human emotions.

ACKNOWLEDGMENTS

This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1548562. Specifically, it used the Bridges system, which is supported by NSF award number ACI-1445606, at the Pittsburgh Supercomputing Center (PSC). "Portions

of) the research in this paper uses the MMIFacial Expression Database collected by Valstar and Pantic" "(Portions of) the research in this paper uses the CK and CK+ Database collected by Cohn and Kanade" "(Portions of) the research in this paper uses the Kaggle Database prepared by Carrier and Courville"

REFERENCES

- [1] François Chollet. 2017. Xception: Deep learning with depthwise separable convolutions. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* 2017-January (2017), 1800–1807. <https://doi.org/10.1109/CVPR.2017.195> arXiv:arXiv:1610.02357v3
- [2] Saeid Fazli, Reza Afrouzian, and Hadi Seyedarabi. 2009. High-performance facial expression recognition using Gabor filter and probabilistic neural network. *Proceedings - 2009 IEEE International Conference on Intelligent Computing and Intelligent Systems, ICIS 2009* 4 (2009), 93–96. <https://doi.org/10.1109/ICICISYS.2009.5357716>
- [3] Richard Gonzalez, Rafael; Woods. 2017. *Digital Image Processing*. Pearson. 1168 pages.
- [4] Ian J. Goodfellow, Dumitru Erhan, Pierre Luc Carrier, Aaron Courville, Mehdi Mirza, Ben Hamner, Will Cukierski, Yichuan Tang, David Thaler, Dong Hyun Lee, Yingbo Zhou, Chetan Ramaiah, Fangxiang Feng, Ruifan Li, Xiaojie Wang, Dimitris Athanasakis, John Shawe-Taylor, Maxim Milakov, John Park, Radu Ionescu, Marius Popescu, Cristian Grozea, James Bergstra, Jingjing Xie, Lukasz Romaszko, Bing Xu, Zhang Chuang, and Yoshua Bengio. 2015. Challenges in representation learning: A report on three machine learning contests. *Neural Networks* 64 (2015), 59–63. <https://doi.org/10.1016/j.neunet.2014.09.005> arXiv:arXiv:1307.0414v1
- [5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2016-December (2016), 770–778. <https://doi.org/10.1109/CVPR.2016.90> arXiv:arXiv:1512.03385v1
- [6] Hayfaa Hussein, Mohsen Naqvi, and Jonathon Chambers. 2017. Study of image-based expression recognition techniques on three recent spontaneous databases. *International Conference on Digital Signal Processing, DSP 2017-Augus* (2017). <https://doi.org/10.1109/ICDSP.2017.8096062>
- [7] Takeo Kanade, Jeffrey F Cohn, and Yingli Tian. 2000. Comprehensive Database for Facial Expression Analysis The Robotics Institute University of Pittsburgh The Robotics Institute The Robotics Institute analysis. In *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition*. IEEE, Grenoble, France, 484–490.
- [8] Patrick Lucey, Jeffrey F. Cohn, Takeo Kanade, Jason Saragih, Zara Ambadar, and Iain Matthews. 2010. The extended Cohn-Kanade dataset (CK+): A complete dataset for action unit and emotion-specified expression. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops, CVPRW 2010*. IEEE, 94–101. <https://doi.org/10.1109/CVPRW.2010.5543262>
- [9] Nicholas A Nystrom, Michael J Levine, Ralph Z Roskies, and J Ray Scott. 2015. Bridges: A Uniquely Flexible HPC Resource for New Communities and Data Analytics. *Proceedings of the 2015 XSEDE Conference on Scientific Advancements Enabled by Enhanced Cyberinfrastructure - XSEDE '15* (2015), 1–8. <https://doi.org/10.1145/2792745.2792775>
- [10] Maja Pantic, Michel Valstar, Ron Rademaker, and Ludo Maat. 2005. Web-based database for facial expression analysis. *IEEE International Conference on Multimedia and Expo, ICME 2005* 2005 (2005), 317–321. <https://doi.org/10.1109/ICME.2005.1521424>
- [11] A. Sada, Y. Kinoshita, S. Shiota, and H. Kiya. 2018. Histogram-Based Image Pre-processing for Machine Learning. In *2018 IEEE 7th Global Conference on Consumer Electronics (GCCE)*. 272–275. <https://doi.org/10.1109/GCCE.2018.8574654>
- [12] Dominic Sayette, M; Cohn, Jeffrey; Wertz, Joan; Perrott, Miachael; Parrott. 2002. A Psychometric Evaluation of the Facial Action Coding System for Assessing Spontaneous Expression Michael A. Sayette, Jeffrey F. Cohn, Joan M. Wertz, Michael A. Perrott, and Dominic J. Parrott University of Pittsburgh. *Journal of Nonverbal Behavior* 25 (2002), 167–186. [papers://8640b87f-578d-4ea3-b2bb-2fcc06c7806/Paper/p6830](https://doi.org/10.1109/8640b87f-578d-4ea3-b2bb-2fcc06c7806/Paper/p6830)
- [13] Ke Shan, Junqi Guo, Wenwan You, Di Lu, and Rongfang Bie. 2017. Automatic facial expression recognition based on a deep convolutional-neural-network structure. *Proceedings - 2017 15th IEEE/ACIS International Conference on Software Engineering Research, Management and Applications, SERA 2017* (2017), 123–128. <https://doi.org/10.1109/SERA.2017.7965717>
- [14] Tian, Ying-li, T Kanade, and J Cohn. 2000. Recognizing LowerFace Action Units for Facial Expression Analysis. *Proceedings of the 4th IEEE International Conference on Automatic Face and Gesture Recognition (FG'00)* 23, 2 (2000), 484–490.
- [15] John Towns, Timothy Cockerill, Maytal Dahan, Ian Foster, Kelly Gaither, Andrew Grimshaw, Victor Hazlewood, Scott Lathrop, Dave Lifka, Gregory D. Peterson, Ralph Roskies, J. Ray Scott, and Nancy Wilkens-Diehr. 2014. XSEDE: Accelerating scientific discovery. *Computing in Science and Engineering* 16, 5 (2014), 62–74. <https://doi.org/10.1109/MCSE.2014.80>
- [16] Michel Valstar and Maja Pantic. 2010. Induced Disgust, Happiness and surprise: an addition to the MMI Facial Expression Database. *Proc Intl Conf Language Resources and Evaluation* (2010), 65–70. <http://www.lrec-conf.org/proceedings/lrec2010/workshops/W24.pdf#page=73><https://ibug.doc.ic.ac.uk/media/uploads/documents/EMOTION-2010-ValstarPantic-CAMERA.pdf>
- [17] P. Viola and M. Jones. 2005. Rapid object detection using a boosted cascade of simple features. (2005), 1–511–I–518. <https://doi.org/10.1109/cvpr.2001.990517>