Oculum afficit: Eye affect recognition

Elmar H. Langholz, Aaron Blythe, Daniel Barker, Zisheng Liao
Department of Computer Science
University of Illinois
Illinois, USA
{elmarhl2, ablythe, dbarker5, zliao13}@illinois.edu

***Abstract*—Recognizing human affect and emotions is a problem that has a wide range of applications within both academia and industry. Affect and emotion recognition within computer vision primarily relies on images of faces. With the prevalence of portable devices, acquiring user images of this nature requires focus, time, and precision. While these systems work great for full frontal faces, they tend to not work so well with partially occluded faces like those of the operator of the device (e.g. smart phones and/or smart glasses) in use. Due to this, we propose a system in which we can accurately infer the overall affect of a person by looking just at the ocular region of an individual.**

***Index Terms*—Affective computing, human affect, eye affect analysis, machine learning, computer vision, convolutional neural network, arousal, valence**

I. Introduction

Using machines to recognize human affect (a concept used in psychology to describe the experience of feeling or emotion) and emotions has always been a challenging topic that amasses huge interest from both academia and industry. There are currently a variety of ways to recognize human affects and emotions such as analysis of body language, voice intonation, and more involved methods like MRI and EEG. However, a more popular and practical approach for affect and emotion recognition is to rely on computer vision by primarily looking at images of faces to analyze facial expressions.

In a world where portable devices [1] have increasingly gained popularity, acquiring a facial image for accurate recognition usually requires full focus, the whole face, and precise timing of capturing the image (capturing a face in transition can result in only a partial facial image). Furthermore, when a subject's face is partially occluded with masks, glasses, objects, or even a beard, the accuracy of recognizing affect and emotions decreases drastically with current models.

We believe it is possible to predict the affect of a user by aiming our attention on just part of their face. More precisely, the ocular region of a subject can convey much about an individual's affect, especially through a portable device.

We chose the ocular region consisting of eyes and eyebrows since as human beings we have learned to evolve to communicate emotions through them [1]. Also, eyes are rarely occluded by users when using portable devices since they must normally look at the screen (close to the camera) in order to capture their image. In fact, smart glasses are used directly in front of the eyes.

[1] Smartphones, VR and/or AR glasses like Oculus Rift, HoloLens, Nintendo Switch, etc...

II. Data set

There exists several data sets corresponding to facial emotion recognition through face images. Some of these are available freely while others need to be requested and consent given for use. While many focus on emotion through a categorical variable, we are interested in affect which can be recorded as a set of numerical continuous variables. As a point in hand, we are interested in the location of the different facial features so that we can focus on the corresponding ocular area which is of interest for this research.

AffectNet [2], as its name suggests, is one of the largest providers of affect and emotion labeled data for a set of face images with a size of 122 GB. It provides approximately one million labeled images which were obtained by querying three different search engines using 1250 emotion keyword in six different languages. It is not freely available and requires consent from the owners.

The data is split in two groups: manually and automatically annotated. For this research we will start by using the manually annotated data which is further split into training and validation sets. Since no test set is provided, we instead use the provided validation data set as our test set and create our own validation data set from the training set by randomly selecting 1% of the entries and setting them aside. We do this to enhance reproducibility and to allow others to perform benchmarking. Table I shows the initial actual sizes regarding how the data set was split.

Affect labels for each face are provided as two separate sets of numerical continuous variables: valence and arousal. While valence corresponds to a sense of how unpleasant/negative to pleasant/positive an event is, arousal focuses on smoothing/calming to exciting/agitating. In both cases, the values have the following range: $[-1, 1]$. Face location, in the form of a bounding box, and 68-point facial landmarks corresponding to mouth, eyes, eyebrows and face contour are provided.

III. Technology and tools

While there are multiple cloud providers, we chose to use Amazon Web Services (AWS) in the West US 2 (Oregon) region [3] due to cost. From it, we made use of Amazon Elastic Block Store (EBS) [4] to store the AffectNet data as well as the derived Oculum Afficit data. Different types of Elastic Compute Cloud's (EC2) [5] were used. A *t2.large* instance was used for preprocessing and a *p3.8xlarge* with four NVIDIA Tesla V100 GPU's was used for training.

The machine learning pipeline developed for this research was written primarily using the Python [6] programming language. The code for it is stored and shared in GitHub [7] within a private organization. For data preprocessing, Apache Spark [8] through PySpark, and backed by Pandas, [9] was used to orchestrate and distribute image processing which leveraged OpenCV [10]. For modeling, training, and evaluation, Keras [11] was used with a Tensorflow [12] back-end along with NumPy [13].

We're also in the process of investigating the use of AWS's Simple Storage Service (S3) [14] and Elastic MapReduce (EMR) [15] to extend our work with Spark into a distributed workload processing system. We may also look at using Lambda as another method for speeding up the preprocessing of the data.

Finally, we use Travis CI [16] to automatically load new code artifacts into an S3 bucket which we will deliver to EC2s for execution.

IV. Methods and design

A. Preprocessing

We derived a new data set by extracting ocular regions from AffectNet. Using the facial landmark points (provided with the data set) corresponding to both eyes and eyebrows and the closest nose point, an initial bounding box comprised of minimum area for these points was calculated for each image. The size of bounding box was proportionally increased by 10% and 25%, horizontally and vertically respectively, while keeping its center point the same. Since the centerline of a face rarely aligns to the horizontal of the image, many of the bounding boxes were rotated. Therefore, the degree of rotation θ was also determined and the image was rotated around the center point of the bounding box before cropping. This is depicted by the top image in Figure 1.

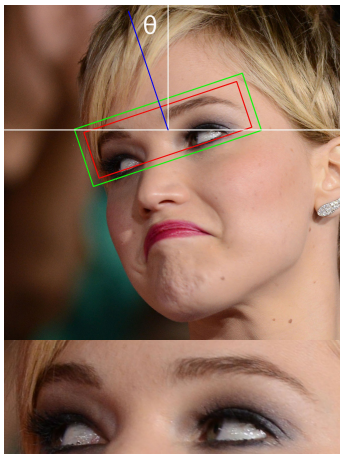

Fig. 1. Preprocessed image example

The first image (top) demonstrates visually the constructed bounding box before and after expanding its size, as well as the calculated rotation θ. The second image (bottom) is the eye slot derived from preprocessing.

A rectangular ocular region image derived through the above process is called an eye slot as shown by the bottom image in Figure 1. If the corresponding arousal or valence is not within the expected documented range, then the image is skipped and not processed. If the extracted eye slot image width and height are not larger than zero or the height is larger than the width, it is also not included as part of the data set. Table I shows the remaining data set size split after preprocessing.

	Training	Validation	Test
Initial	410651	4149	5500
Preprocessed	317521	3218	4500

TABLE I
Data set split

B. Augmentation and normalization

The more and varied the data we train our model with, the better it becomes with respect to accuracy and robustness. One technique used in order to increase the size of the data set as well as introduce perturbations to images is data augmentation. While preprocessing constructs standardized eye slots offline by leveraging facial landmarks, augmentation perturbs and standardizes the eye slot size at run time to construct an infinite amount of different eye slots from one.

During eye slot augmentation different types of perturbations (transformations) were applied to the original eye slot. For each type, a random value within a defined range was generated and used to transform the image. Table II lists the different types of perturbations, value ranges, and corresponding units:

Type	Range	Unit
Brightness	[0.5, 1.5]	Lightness (HLS)
Rotation	[0, 5]	Degrees
Width shift	[0.0, 0.10]	Width percentage
Height shift	[0.0, 0.10]	Height percentage
Shear	[0.0, 0.01]	Radians
Horizontal flip	-	-

TABLE II
Perturbation types, value ranges and units.

Up until this point, we retained the original size of the eye slot derived from the image. However, once the set of perturbations are applied to the image, its size was normalized so that the images could be used with ease for modeling. Since we believe that each individual detail provided within the eye slot is important and relevant to predicting affect, we maintain the aspect ratio of the original eye slot and pad the image at the top and bottom and/or left and right in order to resize it to 520×170 pixels. Figure 2 shows an example of augmentation and normalization used during model training.

Finally, the default image value range of $[0, 255]$ was then normalized to a range of $[0.0, 1.0]$ with the intent to speed up calculations by using floating point operations.

C. Modeling

In computer vision, Deep Convolutional Neural Networks (DCNN) have been working extremely well on achieving state of the art results for multiple different computer vision tasks

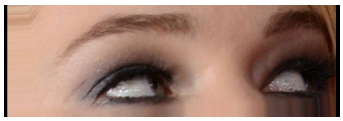

Fig. 2. Augmented and normalized image example

An example of augmentation and normalization of the eye slot from Figure 1. A random set of perturbations were applied and padding is added (in black) to the left and right to maintain the aspect ratio of the image and resize it to 520×170 pixels.

[17], as well as face expression recognition [18]. Due to this, we decided to leverage DCNN architectures to solve a dual regression problem in which our predictor variable was the eye slot images and the response variables (representing affect) were their valence and arousal.

We focused on the VGGNet-like architectures [19], which are comprised of groups called blocks, and implemented the configurations shown in Table III. A block consists of two convolution layers and each convolution used a stride of 3×3. In a block, a convolution layer was followed by a batch normalization layer and a rectified linear unit (ReLU) activation layer. Following a block, a max pool of 2×2 was used. After the blocks, fully connected (or dense) layers of different sizes were added. Finally, at the end of each DCNN configuration, a fully connected layer consisting of two units with a linear activation was added. One for each matching response variable.

DCNN configuration	
M_1	M_2
14 weight layers	15 weight layers

input (512×170 RGB image)	
conv3-16	conv3-64
conv3-16	conv3-64
max pool	
conv3-32	conv3-128
conv3-32	conv3-128
max pool	
conv3-64	conv3-256
conv3-64	conv3-256
max pool	
conv3-128	conv3-512
conv3-128	conv3-512
max pool	
conv3-256	conv3-512
conv3-256	conv3-512
max pool	
conv3-512	conv3-512
conv3-512	conv3-512
max pool	
FC-6144	FC-6144
FC-6144	FC-6144
	FC-2000

TABLE III
VGGNET CONFIGRATIONS

D. Training

Being a dual regression problem (we are trying to learn both valence and arousal using the same filters in one model), the mean squared error loss function was selected to optimize using Adam [20]. We varied the optimizer per run to use different learning rates $\alpha \in \{10^{-3}, 10^{-4}, 10^{-5}\}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. For each run, training was performed with a batch size $\gamma = 32$. While model M_1 was run for $\eta = 35$ epochs, M_2 was run for $\eta = 50$ epochs. Its loss plots are shown in Figure 3 and Figure 4 respectively.

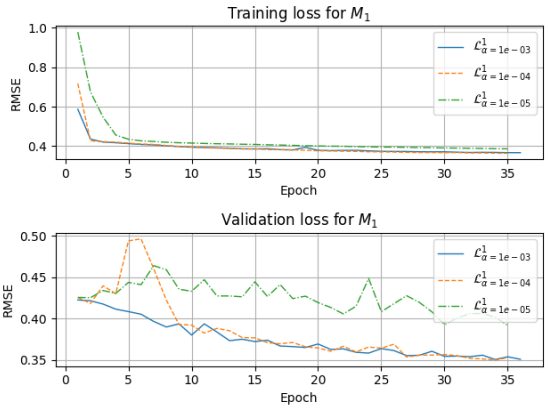

Fig. 3. Model 1 loss plots

The first and second image (from top to bottom) show the loss for the training and validation corresponding to model M_1 which ran for $\eta = 35$ epochs.

Since the validation loss gives us an indication of how well the models can generalize we will focus on it. For M_1, the validation loss plot shows that using an $\alpha = 10^{-5}$ not only behaves erratically, but the loss is minimizes too slow so it does not generalize very well. During the first ten epochs, $\alpha = 10^{-3}$ is minimized faster than $\alpha = 10^{-4}$. Afterwards, they seem to behave similarly but in the end still are minimized slowly.

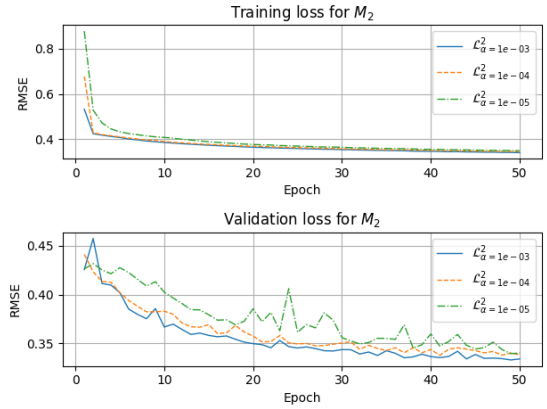

Fig. 4. Model 2 loss plots

The first and second image (from top to bottom) show the loss for the training and validation corresponding to model M_2 which ran for $\eta = 50$ epochs.

For model M_2 we can observe that the validation loss is minimized much faster relative to M_1. All three learning rates behaved in a lot more stable manner. However, we do still see some erratic behavior from $\alpha = 10^{-5}$ which is similar to previously observed. Overall, learning rate $\alpha = 10^{-3}$ provided the best result for generalization across all epochs while $\alpha = 10^{-4}$ followed a close trend to it.

V. Evaluation and results

To evaluate the models, we calculated the root mean squared error (RMSE), Pearson's correlation coefficient (CORR), concordance correlation coefficient (CCC) [21] and sign agreement metric (SAGR) [22] on the test data set. Summarized results for valence (V) and arousal (A) can be found in Table IV for model M_1 and Table V for M_2.

		RMSE		**CORR**		**CCC**		**SAGR**	
	α	V	A	V	A	V	A	V	A
M_1	10^{-3}	**.48**	.42	**.42**	.42	**.34**	.27	**.62**	**.74**
	10^{-4}	.50	**.41**	.41	**.43**	.33	**.28**	.62	.73
	10^{-5}	.49	.42	.34	.35	.23	.22	.60	.72

TABLE IV
Model 1 valence and arousal evaluation

Model M_1 evaluation shows that while $\alpha = 10^{-3}$ performs better on valence, $\alpha = 10^{-4}$ is better on arousal across the metrics except for SAGR. Regarding generalization, our previous observation regarding the close behavior between $\alpha = 10^{-3}$ and $\alpha = 10^{-4}$ is further confirmed since the metrics for each differ in at most 0.02.

		RMSE		**CORR**		**CCC**		**SAGR**	
	α	V	A	V	A	V	A	V	A
M_2	10^{-3}	**.47**	**.40**	**.48**	**.48**	**.39**	**.35**	**.64**	.74
	10^{-4}	.49	.41	.44	.47	.36	.34	.63	.73
	10^{-5}	.50	.40	.44	.46	.35	.35	.61	**.75**

TABLE V
Model 2 valence and arousal evaluation

Unlike with model M_1 the evidence that $\alpha = 10^{-3}$ performs better is a apparent. In this case, valence and arousal are shown to be better across the board, except for SAGR. However, when comparing values between M_1 and M_2 we notice that they do not differ by much. This seems surprising since for the former we trained the model for $\eta = 35$ epochs while for the latter we did for $\eta = 50$ epochs. This gives us an indication that it might be worth exploring M_1 further because it makes use of less weights and therefore its size is smaller.

VI. Discussion

The canonical AffectNet data set is hosted in OneDrive, and we discovered some limitations with the service locking the account we were pulling from for 30 minutes after we downloaded each 5 GB file. It took some time to understand the frequency of the lock/unlock cycle in order to pull the large amount of data we needed into an EBS volume.

Working with a large amount of data that must remain private added some additional challenges for us in sharing with each other for use in our own accounts. We looked into S3 buckets first, but decided it might be too easy to accidentally expose the data publicly. Ultimately, we decided to share an EBS volume as incremental snapshots after key points of processing. The EBS volume contains the raw data in both compressed and uncompressed formats and the initial results of the time consuming pre-processing of the data. The snapshots are only shared with specific AWS accounts in order to maintain security and consistency. We also shared our private S3 bucket which is storing our code and poses less damage in the event of public exposure.

We decided to use GPU-enabled resources for efficiently executing our DCNN architecture for processing the data. AWS puts quotas on these resources, so we had to ask AWS for these quotas to be raised from the default of zero. This is very common in the cloud and acts as both a safeguard for AWS's capacity planning and for the safety of a user's account if a bitcoin miner gets hold of their credentials.

Initially, due to the serial nature of the pre-processing of the images (loading one image at a time and performing cropping and rotation), the process is incredibly slow; Since the data is stored in EBS we can easily attach it to an EC2 instance and start running our scripts, however, it took a total of 6 hours to pre-process both training and validation data. In case we needed to change some part of our pre-processing, we would have to wait another 6 hours for a new run which is not desirable in terms of cost and time efficiencies. Hence, we decided to move on to the Spark framework to parallelize the whole process - specifically we wanted to partition our input file which was a csv file that consisted of the location (filename), valence, arousal, and expression of each input facial image across multiple workers so that each worker could process a sub-portion of the images according to the assigned partitioned input information, for example, which image files to read and the properties of the images.

We first decided to use an EMR (Elastic Map Reduce) cluster that runs the Spark framework automatically. We then needed to decide which storage system we should use. Our options with EMR were only S3 and HDFS, but we discovered both had performance issues with our data:

1. With S3, on each worker we needed to open a connection using boto3 to fetch images for processing, but the speed of downloading the data when we tested it was 300 times slower compared to fetching images directly from a physical disk/hard drive. This could be alleviated if we kept adding more machines into our cluster but at a high cost.

2. HDFS is specialized for dealing with large files but we have a considerable number of small image files that are all smaller than a HDFS block size (64MB), repeatedly reading these small files from HDFS is not efficient in HDFS. What makes it even less attractive is that the HDFS storage on EMR is ephemeral so if the cluster is shut down any data stored on HDFS will not be saved, we would have to keep it running all the time.

We eventually decided to move away from the EMR solution due to the fact that our parallel scheme is not within the scope of traditional map/reduce use cases, we essentially have two inputs, one is referenced by the other (image referenced by csv file). S3 and HDFS would both bottleneck on the data transfer. We would lose the best merit of Spark: **data locality** (image data to be more specific) - with S3 each worker reads from an external source for image data, on the other hand with HDFS the images are not partitioned in the framework

so shuffle is needed. As mentioned earlier, We could have certainly added more machines to our cluster but the cost of running an EMR cluster is not cheap - 2 m3.xlarge instances in our cluster (effectively only 8 cores total) cost about $30 per day.

Our solution here was to use only one EC2 instance, a c4.8xlarge instance which has 36 cores in total and can be easily set up with the Spark framework. This allows us to quickly execute our code built around the Spark framework. The results are shown in Table VI.

Type	Training data	Testing Data
Regular Serialized processing	5hours	2 minutes
Pyspark processing	9 minute 45 seconds	10 seconds

TABLE VI
PYSPARK PROCESSING DATA TIME COMPARISON

We can increase the processing speed even further by scaling up the number of cores in the instance. We've also discussed the potential of using EFS (Elastic File System) across multiple instances to create our own Spark cluster and we hope to test it in future work but within the scope of this project, one powerful instance is more than enough as our results show.

Requesting and setting up the *p3.8xlarge* instance was fast. However, setting up the machine learning tooling to support using the GPU's was not as straight forward. Support for using GPU's required us to follow a long list of instructions for which some required changes in the system (e.g. replacing drivers, removing older versions, etc...) which led to restarting the setup. Furthermore, the cost of use was extremely expensive for training and evaluating the models above: $12.24 per hour. In total we ended up using an instance and its multiple GPU's for 198.438 hours which led to a total cost of $2,428.88. While extremely fast to train our models, we believe that for that cost we could have bought or built dedicated hardware with a single GPU and queued the experiments while still having ample time to finish on time.

In order to have a fully working end-to-end system, we would need to build an eye slot detector as well. While this is currently out of scope, due to having the facial landmarks available, it would make sense to include in place of preprocessing so that we are able to quickly and accurately retrieve eye slots on a live system. We can use the existing data to be able to train it. We hope we can address this in future research.

There are tools on the market now such as AWS Sagemaker where you can simply load a notebook and only pay for the processing time. Further you can select the size and number of machines that would be used for each of the model development phase, the training phase, and deployment of the model with a RESTful endpoint. This would lessen the need for much of the DevOps work that was done for this project. We only learned how to use this sort of option very late in the semester and would run the next iteration on a platform such as this.

Our results were done using a VGG16. We also built a Wide ResNet, however we did not train it based on the costs that we had already been charged training the model using VGG16. In our experience training VGG16 on CIFAR10 took 3 hours, while training a WideRes on CIFAR10 took 16 hrs on a non-cloud based state of the art 2080Ti GPU. This is roughly 5x the amount of time. Since we used around 6 days of processing on the AffectNet with VGG16, we expected it to take a couple weeks to a month and a fairly sizable bill if our original run was any indication. To follow on in this research the WideRes would be the next thing to work on.

VII. Related work

Research focusing on using faces to recognize emotion while leveraging Convolutional Neural Network (CNN) has become prevalent [23] and can even include attention mechanisms to try to address occlusion [24]. Leveraging these techniques using faces to determine emotion and affect on mobile devices [25] also exist. There is also a new focus in combining emotion recognition techniques like EEG and facial landmark localization [26] to solve this problem. Furthermore, generation of facial expression from a neutral expression using Identity-free conditional Generative Adversarial Network (IF-GAN) [27] demonstrates another use following along this area of research. In all the aforementioned, affect is normally treated as a secondary artifact.

While the prior focuses on faces, using the eyes to recognize emotions through non-neural models [28] is something that has been experimented with in the past. However, it is not as prevalent as facial expression recognition. However, as far as we are aware, there isn't any for predicting the affect of a person by focusing just on the ocular region using neural-based models.

VIII. Conclusion

We have demonstrated that it is possible to use the face ocular region in order to infer the affect of a person. Under constraint of the data set in use and the models used, the results show that using the ocular region to predict arousal is more accurate than valence. This seems to make sense if we take into consideration that the mouth is another common conveyor of valence especially when we smile. Having said this, we believe that further research incorporating attention mechanisms might provide value in order to allow the focus on different areas of the ocular region.

Using the Sign Agreement Metric (SAGR) as the key metric. Our model 75% in agreement on the labelled arousal and in the low 60% in agreement on the labelled valence. This is above the agreement between the two human annotators in categorical model of affect of 60.7% in the original paper on AffectNet.

We had many discussions and are coming to the conclusion that for Data Science workloads it is quite a bit different compared against webscale workloads. In many webscale workloads engineers are solving for high availability through load balancing, distribution of compute to multiple resources,

and fault tolerance. For data science workloads, such as the model creation, we seemed to be solving more for get-in and get-out as fast as possible with really strong expensive machinery. Many of the tools are similar in that you still want the repeatability that tools like AWS CloudFormation afford you. However, you only want to use the resources for the amount of time that the processing is actually occurring. There were costs that we incurred both for running time and for forgetting to shut off services. Each time that a service was about to be used we discussed the trade offs in performance (e.g. EBS is more performant than S3 so it's appropriate for processing time, however S3/EBS Snapshots are cheaper, thus better for long term storage when not actually processing.

IX. Division of work

	Work
Dan	Created automation for code delivery into S3. Loaded data into S3 and tested loading it into EMR and Google Cloud Dataproc (never successful). Provided documentation editing.
John	Implemented Pyspark framework for pre-processing image data, EMR cluster exploration including loading data from S3 and HDFS, Performed data processing with Spark with parallel scheme, ML modelings (Wide ResNet), contributed content across the paper
Aaron	Gathered data from research source. Created reusable raw data store on EBS. Pre-processed the data in raw long running form. Created reusable Cloudformation Templates and POC of setting up EMR.
Elmar	Proposed research and implemented ML pipeline: preprocessing, augmentation and normalization, modeling, training, evaluation. Performed training and evaluation of models. Contributed content across the paper.

References

[1] D. H. Lee and A. K. Anderson, "Reading what the mind thinks from how the eye sees," *Psychological science*, vol. 28, no. 4, pp. 494–503, 2017.

[2] A. Mollahosseini, B. Hassani, and M. H. Mahoor, "Affectnet: A database for facial expression, valence, and arousal computing in the wild," *CoRR*, vol. abs/1708.03985, 2017. [Online]. Available: http://arxiv.org/abs/1708.03985

[3] Amazon Web Services, "Amazon web services (aws) - cloud computing services." [Online]. Available: https://aws.amazon.com/

[4] ——, "Amazon elastic block store (ebs) - amazon web services." [Online]. Available: https://aws.amazon.com/ebs/

[5] ——, "Amazon ec2." [Online]. Available: https://aws.amazon.com/ebs/

[6] G. Van Rossum and F. L. Drake Jr, *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[7] GitHub, Inc., "The worlds leading software development platform - github." [Online]. Available: https://github.com/

[8] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin *et al.*, "Apache spark: a unified engine for big data processing," *Communications of the ACM*, vol. 59, no. 11, pp. 56–65, 2016.

[9] W. McKinney, "pandas: a foundational python library for data analysis and statistics," *Python for High Performance and Scientific Computing*, vol. 14, 2011.

[10] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[11] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[12] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[13] T. Oliphant, "NumPy: A guide to NumPy," USA: Trelgol Publishing, 2006–. [Online]. Available: http://www.numpy.org/

[14] Amazon Web Services, "Cloud object storage — store & retrieve data anywhere — amazon simple storage service." [Online]. Available: https://aws.amazon.com/s3/

[15] ——, "Amazon emr - amazon web services." [Online]. Available: https://aws.amazon.com/emr/

[16] Travis CI, GmbH, "Travis ci - test and deploy your code with confidence." [Online]. Available: https://travis-ci.org/

[17] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *CoRR*, vol. abs/1901.06032, 2019. [Online]. Available: http://arxiv.org/abs/1901.06032

[18] S. Li and W. Deng, "Deep facial expression recognition: A survey," *CoRR*, vol. abs/1804.08348, 2018. [Online]. Available: http://arxiv.org/abs/1804.08348

[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015. [Online]. Available: http://arxiv.org/abs/1409.1556

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[21] I. Lawrence and K. Lin, "A concordance correlation coefficient to evaluate reproducibility," *Biometrics*, pp. 255–268, 1989.

[22] M. A. Nicolaou, H. Gunes, and M. Pantic, "Continuous prediction of spontaneous affect from multiple cues and modalities in valence-arousal space," *IEEE Transactions on Affective Computing*, vol. 2, no. 2, pp. 92–105, 2011.

[23] N. Christou and N. Kanojiya, "Human facial expression recognition with convolution neural networks," in *Third International Congress on Information and Communication Technology*. Springer, 2019, pp. 539–545.

[24] Y. Li, J. Zeng, S. Shan, and X. Chen, "Occlusion aware facial expression recognition using cnn with attention mechanism," *IEEE Transactions on Image Processing*, vol. 28, no. 5, pp. 2439–2450, 2019.

[25] C. Hewitt and H. Gunes, "Cnn-based facial affect analysis on mobile devices," *CoRR*, vol. abs/1807.08775, 2018. [Online]. Available: http://arxiv.org/abs/1807.08775

[26] D. Li, Z. Wang, Q. Gao, Y. Song, X. Yu, and C. Wang, "Facial expression recognition based on electroencephalogram and facial landmark localization," *Technology and Health Care*, no. Preprint, pp. 1–15, 2019.

[27] J. Cai, Z. Meng, A. S. Khan, Z. Li, J. O'Reilly, and Y. Tong, "Identity-free facial expression recognition using conditional generative adversarial network," *arXiv preprint arXiv:1903.08051*, 2019.

[28] V. SRamaraj, A. Ravindran, and A. Thirumurugan, "Emotion recognition from human eye expression," *IJRCCT*, vol. 2, no. 4, pp. 158–164, 2013.