Facial Key-point Prediction

Parth Pawar Emily Loh Hongyu Jiang Walid El Baghdadi

I. Introduction

Facial key-point detection has emerged through the years as a crucial task in many machine learning goals and computer vision works. It has become a versatile tool for people in a large variety of situations. It plays a significant role in analyze the emotions on a face, facial recognition for security and biometrics, virtual and augmented reality, accessibility and many other things. The ability to accurately find the points of a face is essential to establishing reliable and useful tools. In this study, we explore knowledge and studies of facial key-point detection and make our discussion as well as produce results in a small experimental attempt to learn flexibility of different machine learning models on the same data set of black and white images.

The significance of facial key-point detection is founded through pinpointing the precise spots of a person's face. By picking out those points, we can gain insight on the face through numbers and calculations (i.e. calculating if the corners of a person's lips are up higher than other coordinates, indicating a smile). Humans work with machines to solve mundane tasks and intriguing challenges just by reading one's face. As technology continues to integrate into our daily lives, these tools of facial feature prediction are becoming more important to enhance individual user experiences and human-computer interactions.

II. Background

A. Problem Statement

As a group, we picked facial feature detection largely due to the real-world application and relevance it held in current events. Working on something directly related to computer science instead of studying data sets of the housing market, popular TV shows, or movie recommendations, it seemed more interesting to touch on the topic of computer vision.

Our question was: given an image of an individual in the color black and white, can machine learning model identify the 15 key points on the face accurately? And we wanted to prove our model could do that. Even though this project is far from being able to tackle real-world challenges, we wanted to learn how this step of the process can lead up to facial recognition, human emotion analysis, medical diagnosis, and many more.

B. Looking at the Data

In key-point analysis, datasets contain those key-points of the face. Most importantly, any successful machine learning model comes from the quality and variety of the dataset. Some cases, colored images are used, while in other cases, the model is set to only understand black and white images. Not only do the colors matter, many models need images of high-quality and a variety of sizes to learn from.

C. Current Detection Technology

Convolutional Neural Networks are often used in this context of machine learning to study the dataset and formulate a model best fitted for the situation. It is well used in the domain of facial keypoint detection and is tailored for image-based tasks. The idea of convolutional layers allows convolutional operations (mathematical operations used that blend or combine two functions to produce the third).

This type of neural network allows for more focus on abstract features and creates a spatial hierarchy between those features that enhance computational efficiency of the network accurately. Fully connected layers of high-level decision making, layers of neurons all connected to each other, generate complex representations of the input data. These layers are what helps predict the coordinates of facial landmarks. Many models utilize CNN architectures for these reasons.

D. Challenges in Facial Keypoint Detection

CNNs can be computational intensive when looking at a lot of data, but, at the same time, these models require large datasets for effective training. If the training was not done well, there may be overfitting due to the small dataset. Regularization techniques must be used wisely to control this significant issue. And, though good with a variety of images, CNN can be affected when data of different lighting conditions, poses, and expressions skew the data too much so there must be careful data preprocessing.

E. Application

The applications of facial key-point detection are everywhere. In human-computer interaction, the ability to identify emotions builds a natural and human-like communication between the computer and individual. In healthcare, there are also uses of facial feature detection to monitor diagnosis of conditions just by analyzing the face. Augmented and virtual reality experiences, use this tool for realistic avatars for an immersive experience. In biometrics, we have a convenient way to identify individuals for transactions, security, access to control and other things. These are just a few of the many technologies that are developing with the use of facial key-point detection.

III. Literature Review

Exploring the dataset involves researching various approaches with different Convolutional Neural Network (CNN) models. It is important to get some adequate research before attempting to make our own model tailored to our dataset. In certain situations, individuals train their own neural networks instead of using the pretrained ones open to the public.

In our particular case, our strategy to finding the model to use was to see what models are being used in studies publicly published. We were found out there was a long possible list of models we could work with such as VGGNet, AlexNet, ResNet, YOLO, GoogLeNet, and LeNet. We decided to narrow it down by doing further research to explore specific kinds of CNNs and pick various versions that have been studied.

A. ResNet

In a published research under IEEE [1], ResNet was used to work on the very same Kaggle dataset [2] we used in our project. This dataset had previously served as the training cases for a competition hosted on the Kaggle platform. The authors of this research used a specific model: the Masked Loss Residual Convolutional Neural Network (ML-ResNet) which was structured to navigate around the challenges posed by the missing values of the dataset.

Because of the unconventional nature of the dataset, they wanted to avoid a skewed and biased result with the images missing some coordinate values. The ML-ResNet was engineered to tackle these challenges with the masked loss function. The function accounts for every mini batch of data and help prevent overfitting of the model due to missing values. This was to improve generalization of the model. They also applied a transformation of the dataset of images. By flipping the images horizontally and changing the coordinates accordingly, they doubled the amount of training sample to use on their model. They proved that with masked loss function, the CNN used becomes far more accurate when dealing with missing values. Since they tried ResNet, it seemed like a good idea for us to do it as well.

B. AlexNet

This research looked at for AlexNet was also done under IEEE [3]. The researchers used this model after learning about transfer learning in the domain of machine learning. The pretrained model cannot simply be used optimally on the dataset for this new task. These researchers found out they could make the learning process of this model even better by assigning the original pretrained model to a new task to perform and build off of that using the method of ”transfer” learning. With this technique, they planned to make the model fast and efficient with precision.

They trained and tested their AlexNet model of twenty-five layers on the ORL (Olivetti Research Laboratory) and CUHK (Chinese University of Hong Kong) datasets. They managed to get good and high accuracy on the results from their model. They managed to prove that transfer learning was indeed effective on recognizing faces by learning features and then working with a different task. They managed to reach an accuracy rate of 98% and 99% as well as a low variance of 0.03 with the ORL and CUHK datasets using fine-tuned hyperparameter AlexNet. We decided to try out AlexNet as well.

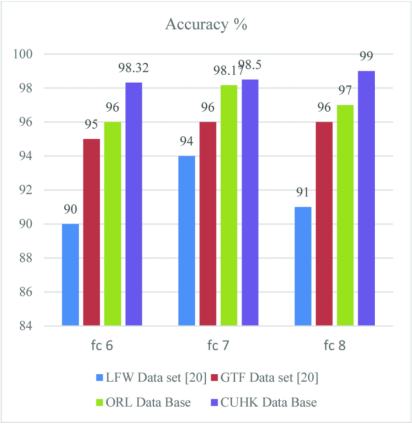

Fig. 1. Graphic of accuracy on the ORL and CUHK datasets

C. YOLO

In this study, the research published under IEEE worked on YOLOv3 [3] as a deep face detector as the researchers found that this model provided real-time detection and is optimized well for speed. There was also end-to-end training which allowed the model to be trained in a simple way, directly mapping the inputs to the outputs. The convolutional network of 106 layers used a pyramid network which captures complex features for varying poses, sizes, and lighting. They also built this with a minimizing loss function and penalties for missing detection.

(a) Occlusion and Scale

(b) Pose

Fig. 2. Images used to test the YOLOv3

Over the course of training through 480 epochs, the researchers evaluated their model on the WIDER FACE data set of more than thirty-two thousand images with numerous different attributes made for a variety of poses, lighting, and scales. The dataset was of photos filled with many people, detecing more than just one face but all the faces in the frame. In the end, the researchers managed to take the YOLOv3 originally trained on COCO dataset to work on their dataset. They managed to achieve speed while receiving good precision

data curves. This made YOLOv3 as a good model to try out on our project.

IV. Data Description & Exploratory Data Analysis

A. Understanding Given Data

Given by Biginepalli on Kaggle [2] there are 7049 images under the training images folder and 1783 images for testing. In the case of our project, it was important to have the actual values to compare to the prediction values our models were generating so we could find out how accurate the models were. To do this, we had to consider dropping the extra 1783 images to focus on just using the 7049 images that already have data for plotted keypoints. This can be considered as a small sample set for training as most deep neural networks would need a lot of images to fully learn the pattern of the coordinates.

The dataset also came with a comma-separated values file called training.csv and this file included thousands of values for different coordinates: x-y coordinates for the center left and right eyes, x-y coordinates for the inner and outer corners of the left and right eyes, x-y coordinates for the inner and outer ends of the left and right eyebrows, x-y coordinates for the tip of the nose, x-y coordinates for the left and right corners of the mouth, and x-y coordinates for the center of the top and bottom lips. With 15 different features and 30 coordinates, the dataset provided a simple mapping of important features on the face.

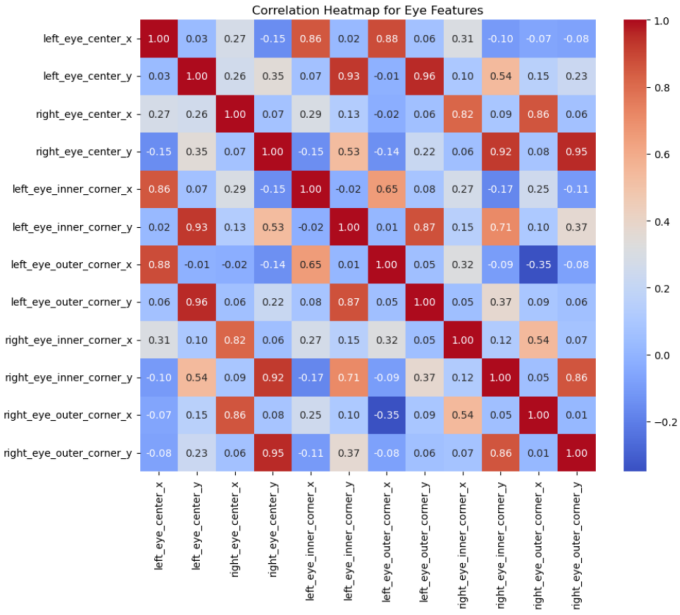

Fig. 3. Heat map on the coordinates of the eye. High correlation shown between y values of the eye

B. Exploratory Data Analysis

Doing exploratory data analysis was difficult due to the nature of the dataset. Because we weren't looking at data that showed patterns like cause and effect, it was hard to see direct correlation or figure out what attribute of the data can be dropped if causing a wave of outliers or was irrelevant to the data. Traditional plots like histograms, scatter plots, pair plots, etc. couldn't easily capture the complexity of facial keypoint data.

One stand out result was simply the fact that most y-coordinates on both sides of the eye, mouth, eyebrows, and, in general, face were similar due to the fact that many images were aligned to be facing straight at the camera. Though seeing a striking red on a heat-map could signify a revelation about the dataset, it is common knowledge that faces are mostly symmetric and so similar y-values shouldn't strike us as something new. Especially of the eyes where most people have aligned eyes that are parallel to the ears, hence, almost draw a straight line across the face.

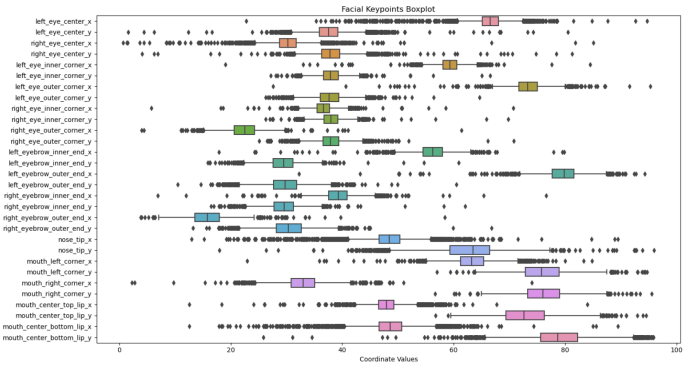

Fig. 4. Lengthy box plot of all the features

The coordinates represent spatial relationships on the face and, while some plots can show relationships between two variables, it may not provide a good representation and explanation of the arrangement of keypoints. Also many images included various different angles and poses, causing a lot of the data to have outliers due to the placement of a face and the features on the image. There were too many points to consider and the outliers that were generated due to images of off-centered faces like shown in "Fig.4" where the box plot isn't helping on the large data and numerous outlier points. Many images were also missing points on the dataset and it was quite nearly impossible to go through the set ourselves to fix the problem.

C. Working with Missing Values

A large problem arose when we reached analyzing the data and splitting it for the models. Majority of the data did not include all the keypoints needed on the face especially the corners of the eyes, eyebrows, and mouth. Over 69% of the images weren't entirely looked through for points so it was difficult to work around this. There are many methods to "fixing" this hole in the data.

We tried using the median or mean to replace those values or just completely dropped the data. Previous research we did mentioned using a way to mask the loss of the dataset but it was difficult to follow through without more knowledge of machine learning. Using the median, mean, or dropping values

would drastically change the structure of our project and affect the results of our models. It was a learning process to try to reach the best model with the knowledge from class and the research done outside of school hours.

V. Proposed Methodology

Our methodological approach involves a comprehensive exploration of deep learning models, strategically testing out three different models to make use of their unique strengths. The selection enabled us to try different things and explore our options within the context of our research objectives. The specific models we tested out were ResNet50, AlexNet, and YOLOv3.

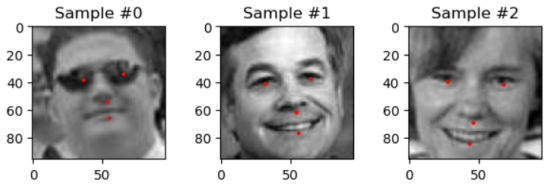

Fig. 5. Small sample of incompletely mapped faces

Like mentioned previously, there are missing values in the dataset. For deep learning, it is of utmost importance that the dataset is large enough to train the model well. In this case, with over 7000 photos, we can get a lot of progress with our model. To handle the missing values in the dataset, we performed data preprocessing by replacing the missing values with the median of each respective feature. While this produce some downsides, such as overfitting or difficulty identifying with images of faces at different angles, it was better than throwing away around 5000 test cases. Dropping so many training images would have a significant negative impact on the size of the dataset while we need a large training dataset given the nature of deep learning.

A. ResNet50 Implementation

For the ResNet50 model, we focused on using the characteristic of transfer learning to apply the model to our task. We researched the ResNet50 Keras architecture and used that. It had been trained through the ImageNet database. We then adjusted the architecture accordingly by choosing the 96, 96 input size. The foundation of our model included the ReLu function before applying the ResNet50 preset model. For this model, we chose an arbitrary smaller dropout value to avoid any chance of overfitting. Then we included the dense layer to classify the images accordingly to the 30 different x-y coordinate points.

Hyper parameters such as the learning rate and early stopping were crucial to control the model's convergence and the chances of overfitting. These were also included in other models. Since it was essential, we included these parameters in the ResNet50 implementation as well. This model was then trained for 100 epochs with our Kaggle train data on the data split of 80:20 of the 7049 faces for about 20 minutes of wait time (for the case of TP4 GPU).

B. AlexNet Implementation

The AlexNet model, implemented using TensorFlow and Keras for this project, features an architecture with five convolutional layers having filter sizes of 96, 256, and 384, and varied kernel dimensions. Each convolutional layer is followed by batch normalization, with max pooling layers interspersed to reduce spatial dimensions. The network then transitions into fully connected layers, with the first two featuring 4096 neurons each, ReLU activation, dropout of 0.5, and batch normalization, strategically designed to prevent overfitting.

The data preparation process is noteworthy. Images originally sized at 96x96 are resized to 227x227, aligning with the standard input size for AlexNet. This resizing is crucial for compatibility with the network structure and for effective learning. The dataset is split into 80:20, facilitating a comprehensive training regime.

The model employs the Adam optimizer with a learning rate of 1e-2 and uses mean squared error as the loss function. Techniques like EarlyStopping and ReduceLROnPlateau are incorporated to optimize the training process, preventing overfitting and ensuring efficient convergence. The model is trained over 100 epochs, with the training process dynamically adapted to the available hardware, whether TPU, GPU, or CPU.

C. YOLOv3 Implementation

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Fig. 6. Structure of Darknet-53

YOLOv3 (You Only Look Once) is a very powerful model used for image recognition and feature detection. Using the Darknet-53 as its backbone and feature extractor, YOLOv3 receives input of 3-layer images (RGB).

For Darknet-53 itself, several residual blocks are used to build its main structure. Each residual block consists of two convolutional layers, with the use of batch normalization and ReLU as activation function. After being preprocessed by Darknet-53, a single image would be put into three different feature maps with size $13 \times 13 \times 255$, $26 \times 26 \times 255$, $52 \times 52 \times 255$.

In this project, the YOLOv3 model is attempting to receive images with size $96 \times 96 \times 3$. However, since our images are grayscale images with size $96 \times 96 \times 1$, during the image preprocessing step, we first converted them into RGB images by simply copying and pasting the information on the first layer to the rest two layers. Then we normalized the images so that they could be used for training. Due to the different input sizes, the sizes of feature maps are adjusted to $3\times3\times255$, $6 \times 6 \times 255$, $12 \times 12 \times 255$ accordingly.

D. Choosing the Best Model

Our proposed methodology came down to being able to confirm which model we wanted to stick with as the most accurate for discussion. We wanted to be able to differentiate which model did what best and clearly compare which model has achieved the goal in the most efficient way. To make this happen, we took the finish models and used the test images on them. This would produce mapped images that we could briefly see how well each model did and, at the same time, how different the results were. Using graphics and some math calculations like the mean squared error, we planned to find out which model we wanted to finalize with.

VI. Experimental Results

Through our experiment of several different models, we got to see what they resulted in and saw some interesting comparisons between the three models. We used different graphs to show what happened during the training and the mean squared error to compare which model did the most precise calculations during their predictions.

A. ResNet50 Results

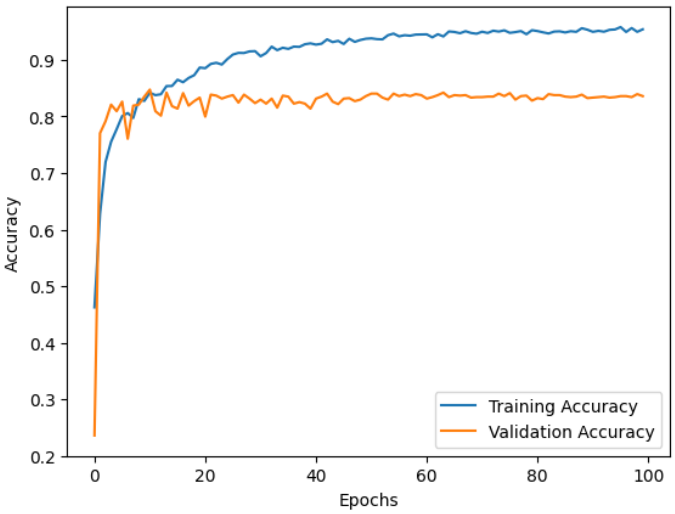

Fig. 7. Graphed representation of the accuracies of the ResNet50 model

After running the ResNet model through training for 100 epochs, we noted down the values of loss and accuracy throughout as well as learning rate. With a high training accuracy of 0.9541, we know that the model is fitting really well to the data. The model is performing good on the given training dataset. The ratio to the total number of instances with correctly predicting became increasingly high at the near end. On the other hand, the validation accuracy was 0.8360 that showed the model generalized decently well to new, unseen data. When graphed, we found out that it was consistently around the same validation accuracy for many epochs and was stuck at that level with little to no improvement. Looking the figure of the accuracy, we can see there is about a 0.1 difference, hinting at some overfitting of the model.

A large portion of the first few epochs showed that there was a large training loss and lots of validation loss mostly due to the model taking time to adjust to the training data and being able to start generalizing more accurately. This pattern slows down to a low loss on both training and validation as the model starts to fit better with unseen data. Epochs 50 and higher show that the loss values are generally constant and do not change too much.

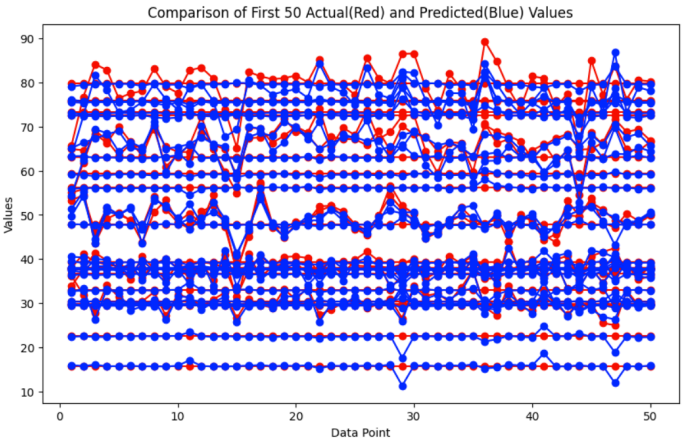

Fig. 8. Visual for MSE on the test data from the 80:20 split

The mean squared error for this model was around 1.82 which is, not perfect, but close enough to get good predictions especially for easy to identify faces. This is not considering how flawed the dataset turned out to be. On some facial features, there was definitely more inconsistencies than other parts of the face.

Other values such as the mean absolute error and the changing learning rate also played a part in showing what happened during training. The mean absolute error for validation was fluctuating a lot, showing its inconsistencies early on in the training process. As it went on, the results started to show that less overfitting was happening after 60 epochs. Meanwhile, the changes of the learning rate slowed down around the same time. It started off with the learning rate of 0.001 but exponentially became smaller to the value of 4.7476e-06.

In the case of this model, there was no early stopping used and the training did complete all 100 epochs. This could mean the validation loss could still be improved or that we have a chance at trying a longer run with more epochs to see if there is more that can be shown.

B. AlexNet Results

The AlexNet model's performance in facial keypoint detection over the course of 69 epochs offers a comprehensive view of its learning trajectory and overall improvement. Initially, the model exhibited a high training loss and an even higher validation loss, indicating a significant discrepancy between training and validation performance.

As training progressed forward, there was a notable improvement in both the training and validation metrics. The training loss showed a dramatic decrease, particularly in the early epochs. This rapid initial decrease in training loss indicates the model's quick adaptation and learning capabilities. Concurrently, the validation loss, which initially spiked, began to decrease steadily, closing the gap with the training loss. This reduction in validation loss is a crucial indicator of the model's improving ability to generalize to unseen data.

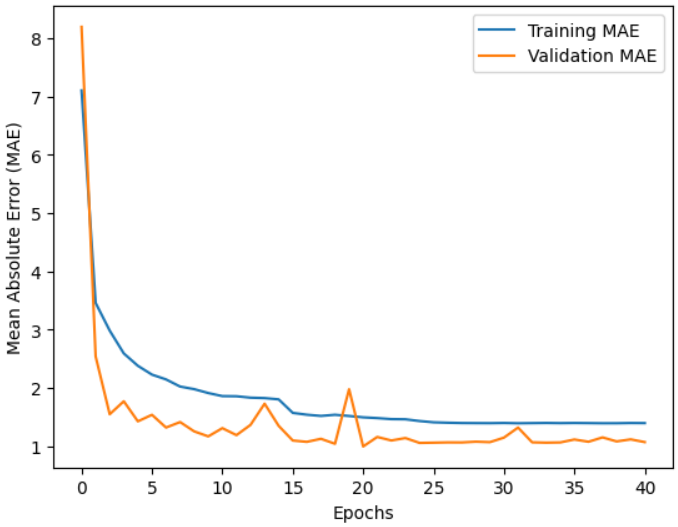

Fig. 9. Graphed representation of the mean absolute error of the model

The mean absolute error (MAE) on both training and validation data followed a similar downward trend. By the 41st epoch, the training MAE had gone all the way down to the value of 1.3938, and the validation MAE to 1.0674. These MAE values are particularly important as they directly reflect the model's accuracy in predicting facial keypoints – the lower the MAE, the more precise the predictions.

The learning rate adjustments, implemented through the ReduceLROnPlateau strategy, played a vital role in this improvement. The model started with a higher learning rate of 0.01, which was reduced in subsequent epochs when the validation loss plateaued. These reductions in the learning rate helped prevent the model from overshooting the minimum loss and ensured more stable and gradual improvements in performance.

The employment of EarlyStopping was another critical aspect of the training process. This strategy, which stopped the training once the validation loss ceased to improve, was instrumental in preventing over training and potential overfitting. The training was halted at the 41st epoch, with the model weights being restored to those of the epoch where the best validation loss was observed.

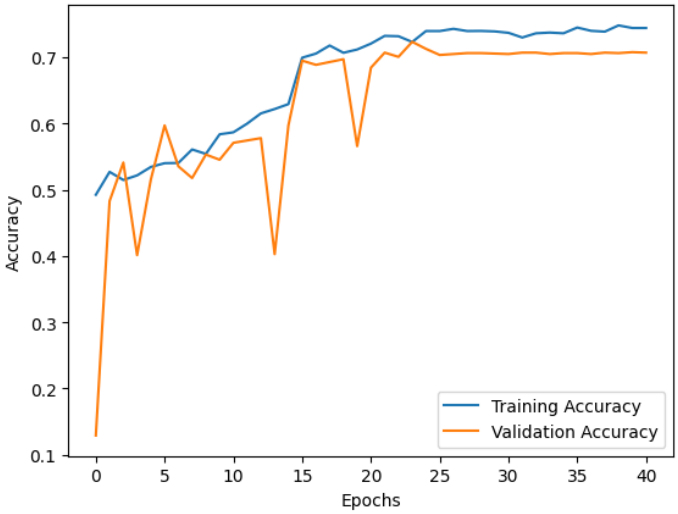

Fig. 10. Graphed representation of the accuracies of the AlexNet model

The model managed to come up with an accurate enough model to compare to the original data points. The generated difference of the MSE was 3.51 and the training accuracy reached a decent value of 0.7443 and a validation accuracy of 0.7071 which are similar values.

C. YOLOv3 Results

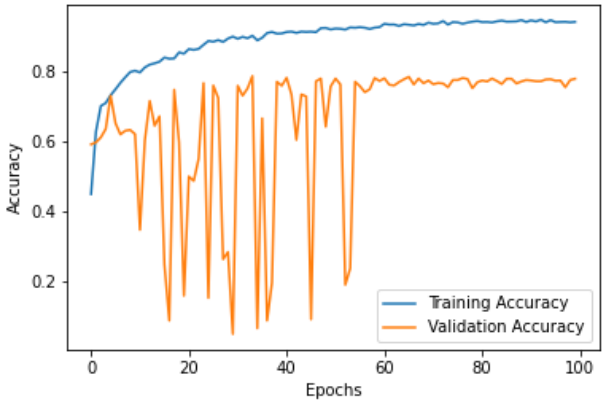

Fig. 11. Graphed representation of the accuracies of the YOLOv3 model

After running the YOLOv3 model through training for 100 epochs, we noted down the values of loss and accuracy throughout as well as learning rate. With a high training accuracy of 0.9339, we know that the model is also fitting really well to the data. The model is performing good on the given training dataset. However, when we used the model to do the prediction, it showed a relatively high MSE of 2.179.

There are several reasons for that. YOLOv3, in most cases, is used to process RGB images with three layers. During the image preprocessing step, even though we converted our gray-scale images into RGB images, the information contained in these images are not as abundant as RGB ones. Thus, feeding and predicting with gray-scale images generated a relatively high MSE score. We guess if we are using RGB images for training and testing, the performance of YOLOv3 would be better than that of ResNet50. At the same time, during the training process, we found that a huge part of our dataset was missing. In order to make sure that our training process could proceed well, we replaced the missing places with the median value of the existing data. This might confuse the model while training. Also, unlike the other two models, during the training process, YOLOv3 did not apply any dropout algorithm. This might cause the model to be slightly overfitting.

There are some ways to be considered to improve the performance. First, switching to training and testing with native RGB images, which YOLOv3 is better suited for, can be more effective. For handling incomplete data, exploring alternatives to median value imputation, or acquiring a more comprehensive dataset could prove beneficial. Implementing dropout in the training phase can help prevent overfitting.

VII. Discussion

A. What Went Right

We managed to find a model that could work well at high accuracy and a good mean squared error value. It was hard to find out which models to try out. Because these models were all basically good for transfer learning and, with just some good minor modifications, we could make use of convolutional neural networks and use it on our current task.

Fig. 12. Generated accurate predictions from ResNet50

Due to ResNet50's 50 layers and highly trained structure, it was very convenient and applicable to work with transfer learning. It was definitely a large worry when working with this dataset that we wouldn't be able to produce a decent model due to the Kaggle dataset missing a lot of values. It is a relief, not only did ResNet50 work, but, each to their own, the models all were good at something different for the images.

B. What Went Wrong

The dataset was definitely very tricky to work with despite the fact that it looked good at first glance. The huge dilemma about picking whether to drop or keep the cases with missing values was very important and significantly impacted the way our results panned out. Because we decided to stick with this dataset, a lot of errors stemmed from the missing values. Even though the MSE may be low for ResNet50, printing out some examples may show that the points aren't on the eyes or other facial features. This is because the median values that filled the missing value slots are incorrect in the first place. It could be said the model wasn't trained correctly at all due to the dataset being incomplete.

There were a lot of obstacles when trying to adjust for the missing values and it was difficulty to run the model to retrain often because of how long it would take sometimes. Because of how large scale the dataset was and the nature of the models, it was not easy nor convenient to be constantly trying for new values in our parameters, dropout rate, etc. and readjust for the missing values.

C. What to Change

ResNet50 did a good job and we wouldn't want to change that. We could try things out with a better dataset or follow something similar to the ML-ResNet which did a great job at considering the missing values. With that, we can definitely achieve higher accuracy and be able to claim that all 7049 entries helped trained the model to make it better.

It would also be good to learn how the other two models could apply such as being able to work with RGB value images or images of faces in different angles or sizes. There is a lot more to facial keypoint detection than just a portrait 1:1 ratio image in grayscale. In reality, technology is pursuing better models that can take care of more than a rigid requirement of an image.

But, just for the focus of this dataset, it would be good to attempt the ML-ResNet and exercise more time on testing out different dropout rates, other additional layers to the model, changes in training and test split, as well as better analysis of the given dataset.

D. Application

Though our model may seem weak compared to prior studies by students and researchers of higher expertise, the ResNet50, YOLOv3, and AlexNet can all be reapplied to future projects of similar task when it comes to teaching the computer how to read images and deduce something significant from them.

Such models are already applied in the field but more research can be done on our end to learn about how flexible a convolutional neural network model can be regarding the type of photo input it receives. We can move from a square gray-scale photo with the faces centered to more complex images of multiple people in color and away from the center of the graphic.

VIII. Conclusion

In our research of deep learning models for facial keypoint detection, ResNet50 stood out as the best model of the three including AlexNet and YOLOv3. Despite our dataset from Kaggle being full of missing values, ResNet50 was able to generalize well and show some form of accuracy.

The 100 epoch training of the model resulted in a small mean squared error of approximately 1.82, putting it at first place for smallest MSE value of the three models. It also had the best accuracy values throughout its epochs, setting itself at 0.9541 training accuracy and 0.8360 validation accuracy.

There was some signs of overfitting due to the nature of the dataset but the model showcased effective generalization to new, unseen data throughout. The model proved to be adaptive and resilient to the task at hand and worked through transfer learning well. The model's accuracy made it the finalized choice for our project.

In the future, considerations for a better dataset or finding a way to work around the missing values would approach even greater accuracy for this task and find more real-world applicability.

References

[1] Shaoen Wu, Junhong Xu, Shangyue Zhu, Hanqing Guo, A deep residual convolutional neural network for facial keypoint detection with missing labels, Signal Processing, Volume 144, 2018, Pages 384-391, ISSN 0165-1684, https://doi.org/10.1016/j.sigpro.2017.11.003.

[2] Nagasai Biginepalli, facial keypoint detection, https://www.kaggle.com/datasets/nagasai524/facial-keypoint-detection/data

[3] G. Lokku, G. H. Reddy and M. N. G. Prasad, "A Robust Face Recognition model using Deep Transfer Metric Learning built on AlexNet Convolutional Neural Network," 2021 International Conference on Communication, Control and Information Sciences (ICCISc), Idukki, India, 2021, pp. 1-6, doi: 10.1109/ICCISc52257.2021.9484935.

[4] F. Gurkan, B. Sagman and B. Gunsel, "YOLOv3 as a Deep Face Detector," 2019 11th International Conference on Electrical and Electronics Engineering (ELECO), Bursa, Turkey, 2019, pp. 605-609, doi: 10.23919/ELECO47770.2019.8990641.

[5] J. Redmon and A. Farhadi, "Yolov3: An Incremental Improvement," Arxiv.Org, University of Washington, 2018. [Online]. Available: https://arxiv.org/pdf/1804.02767v1.pdf. [Accessed Dec. 2, 2023].