My Reaction Says it All

Garrard, Jeremy
jeremy.garrard@gatech.edu

O'Donnell, Liam
lodonnell32@gatech.edu

Taskovski, Kelly
tlee459@gatech.edu

Abstract

In digital communication, people can miss the emotional cues that help us understand each other in face-to-face conversations. This project explores how facial expressions can be used to generate responses text-based messaging. By combining emotion detection from facial images with the context of a message, our system suggests a response to convey the emotion of the individual receiving it. We provide a convolutional neural network to capture emotion, and a fine tuned seq2seq model for text generation, making conversations feel more human—even through a screen.

1. Introduction

A large part of human communication is nonverbal [4], and our goal is to harness this in the context of digital interactions. This projects focuses on on analyzing facial expressions to generate a response to enhance communication.

We will pair detected emotions with a received message (context). By combining these three elements, we can infer how someone might feel respond when receiving a message based on their facial expression.

Using this information, our system will generate responses that convey the emotion that user is feeling.

For example, imagine receiving a text from your mother asking if you will be careful on a trip for the hundredth time. You might feel slightly annoyed and either ignore the message or respond hastily. Our system will detect this frustration, interpret the context, and draft a response for your approval.

Today, some apps and tools can scan the face of a person and guess how they are feeling, such as happy, sad, or angry. Others focus only on the words in a message to understand the emotion.

These tools usually stop at naming the emotion and do not do much to help the person convey the message. Many people struggle to express themselves clearly over text, especially when emotions are involved. This can lead to misunderstandings, hurt feelings, or arguments. Our implementation attempts to convey the feelings of the user into a concise text message.

2. Facial Emotion Recognition

2.1. Overview

We chose to use a convolutional neural network (CNN) to classify emotions based on its well-documented ability to capture spatial patterns in images—particularly facial features such as the eyes, mouth, and brow—that are essential for recognizing emotional expressions.

Our initial model was intentionally simple- to establish a baseline. Over time, we increased its capacity and complexity to improve accuracy. We also focused on hyperparameter tuning and experimented with different loss functions and class weighting strategies to better capture underrepresented emotion classes.

We specifically chose not to start from already pretuned models like VIT as we thought experimentation as we thought experimentation with our own architecture would give us greater insight into how specific design decisions affected performance.

2.2. Facial Emotion Dataset

We used the FER-2013 dataset which was first used in a challenged hosted by Google for emotion recognition [1]. The winner of this challenge had an accuracy of 70.22% using a blend of sparse filtering, random forests, and SVM. The idea was to use combinations of ML techniques to approach accuracy of complex ideas.

The dataset contains 35,887 grayscale images of faces, each sized 48x48 pixels and labeled with one of seven emotion categories: Angry, Disgust, Fear, Happy, Sad, Surprise, and Neutral.

The dataset was created by collecting facial images from the internet using the Google image search API, followed by human labeling. It does not have an even class distribution, with angry, disgust and fear notably having a much smaller dataset in total than other classes, as shown in

Figure 1. The uneven class distribution will lead directly to model architecture and tuning experiments.

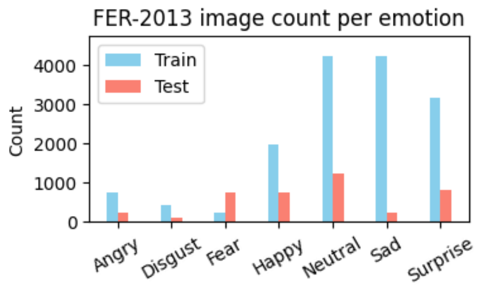

Figure 1. FER-2013 image count per emotion

Some class labels are questionable, though we included as-is in the dataset. For instance, Figure 2 shows some images in the FER-2013 dataset that are labeled as *fear*, though they may be more accurately classified under a different emotion.

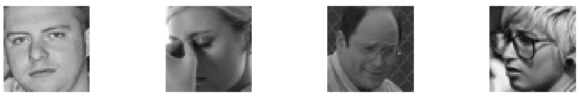

Figure 2. Sample training images labeled as *fear*

2.3. Initial CNN Setup

Our baseline model was a simple convolutional neural network composed of three convolutional layers followed by a fully connected output stage. Details are provided in Table 1.

Layer	Output Shape	Details
Input	1×48×48	Grayscale image
Conv1 + ReLU + MaxPool	32×24×24	3x3 conv, stride 1, pool 2x2
Conv2 + ReLU + MaxPool	64×12×12	3x3 conv, stride 1, pool 2x2
Conv3 + ReLU + MaxPool	128×6×6	3x3 conv, stride 1, pool 2x2
Flatten	4608	–
FC1 + ReLU + Dropout	256	Dropout p=0.25
Output (FC2)	7	Emotion class logits

Table 1. Initial CNN architecture

As shown in Figure 3, this model captured a 55% accuracy in the test dataset but when looking at our accuracy curve it was clear the model was overfitting our data.

Another clear issue confirmed by our confusion matrix in Figure 4 is that we had a class imbalance problem. Class recall for happiness (3 in the below) is 76% while fear was a lowly 42%.

2.4. Loss Considerations

Our initial model was implemented with a simple cross entropy loss. This loss mechanism, when confronted with large datasets for an emotion, like happiness, it performs perfectly well, with our class recall breaking the overall accuracy score of the 70.22% contest winner. However lower represented classes are sort of ignored by CE because

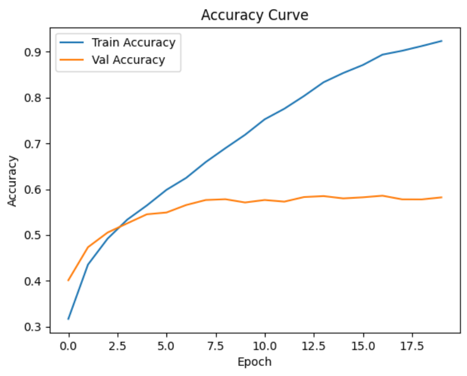

Figure 3. model 1.0 accuracy curve

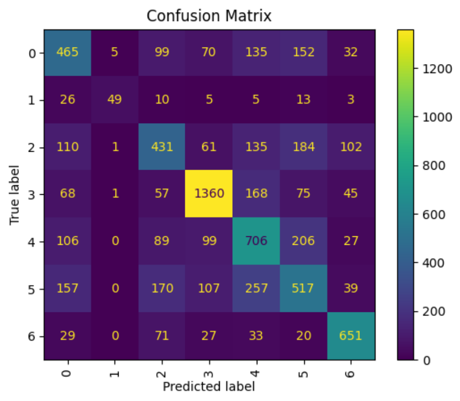

Figure 4. model 1.0 confusion matrix

the overall loss average isn't impacted by a large number of missed class predictions.

In order to remedy we ran a heads up comparison between two variants of losses- weighted cross entropy and focal loss. We implemented weights for CE as the normalized inverse of frequency, so weights sum to 1 but are based on 1/count for each class. We also implemented focal loss with a sweep search for the best gamma. What we found was that accuracy scores just happened to be better with weighted cross entropy vs focal loss.

We hypothosized that, focal loss, especially at higher gammas, started to emphasize training on unrepresentative images for small classes like fear, as shown before, and therefore hurt overall accuracy. We think weighted CE struck a better balance at accuracy and generalization between classes.

Figure 5 illustrates what we think was happening, where FC very heavily punishes high confidence misclasses, where in this dataset that might actually be the correct classing overall.

2.5. Architecture and Hyperparameter Tuning

When considering next steps for our model we focused on the effect of capacity and hyperparame-

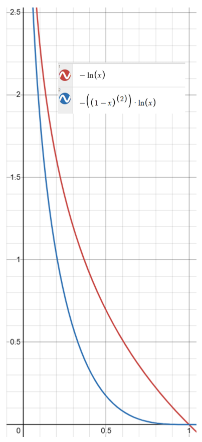

Figure 5. CE (red) vs Focal Loss (gamma=2)

ters when it comes to the main challenge of this dataset - overfitting. Our initial thought was to increase capacity and dropout at the same time to encourage a stronger, more generalized model but my loss divergence showed a surprising thing- increasing model strength actually caused faster and faster plateauing of the validation loss while train loss continued to decrease.

We started to think differently about capacity needs- we believe a simpler model, not a complex one, would be better for generalization. We hypothesized that higher capacity models were starting to learn more 'noise' than useful features and resulted in messy training. We also introduced data augmentation in the dataloading step to force a more generalized look at our somewhat small data set.

Another key consideration was hyperparameter tuning. We focused on batch size, learning rate, and dropout. Intuitively, we figured that using a smaller batch size — scaled to the learning rate — might help reduce overfitting. Our thinking was that this would result in more 'cautious' weight updates, even if it meant slower training overall. As for dropout, we knew it required some balance, but we relied on our training protocol, Optuna, to determine the best setting based on performance metrics.

2.6. Emotion Recognition Evaluation and Best Result

Our final model architecture ended up having less parameters than our inital setup, at 851k as opposed to the initial 1.2mm. Details are provided in Table 2. However, ended performing at 66% accuracy, as shown in Figure 6. The main contributors to our success is our increased emphasis on tools to force generalization were adding batch normalization to stabilize training and accelerating convergence, replacing ReLU with LeakyReLU, utilizing balanced dropout across both convolutional and fully connected layers to reduce overfitting without disrupting gradient flow. Additionally, we implemented data augmentation to improve generalization across small cases and tuned every critical hyperparameter using Optuna's structured search.

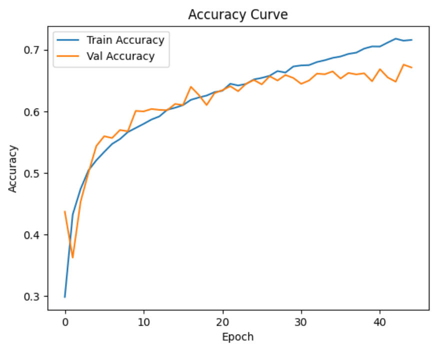

Figure 6. Final CNN Model Accuracy

Final CNN Architecture (851k parameters)
Input: 1×48×48 (Grayscale)
Conv Block 1 Conv2D (1→20), 3x3, padding=1 BatchNorm2d, LeakyReLU Conv2D (20→20), 3x3, padding=1 BatchNorm2d, LeakyReLU MaxPool2d (2x2), Dropout Output: 20×24×24
Conv Block 2 Conv2D (20→40), 3x3 BatchNorm2d, LeakyReLU Conv2D (40→40), 3x3 BatchNorm2d, LeakyReLU MaxPool2d (2x2), Dropout Output: 40×12×12
Conv Block 3 Conv2D (40→80), 3x3 BatchNorm2d, LeakyReLU Conv2D (80→80), 3x3 BatchNorm2d, LeakyReLU MaxPool2d (2x2), Dropout Output: 80×6×6
Flatten: 2880 FC1: Linear (2880→256), LeakyReLU, Dropout FC2: Linear (256→7)

Table 2. Compact Representation of Final CNN Model

One point of contention we still have is with the dataset we are using. We truly believe there are some misclasses in our dataset and, as a result of our strict weighted loss function, our model is learning some unhelpful patterns which decreases overall accuracy. If we had the ability, going after a well vetted data set 10x bigger would result in a much better accuracy score. To drive this point home see Figure 7 for a confusion matrix that takes a random sample picture from each type of error. We believe some of the images labeled as surprise are actually fairly all over the place- from disappointed to sad to overwhelmed.

This also raises the question of whether model misclassifications sometimes reflect ambiguity in human emotion perception, rather than model failure.

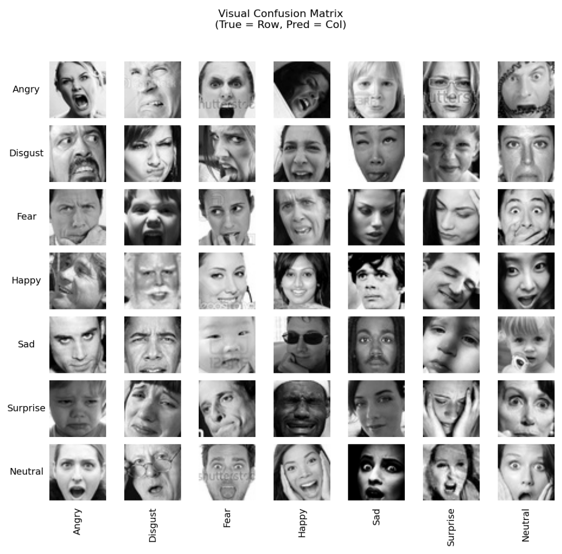

Figure 7. Final Model Accuracy

3. Response Generation

3.1. Overview

The second phase of our project was to use the prediction facial reaction to generate a response to a given text message given the message context. The probability model for this use-case was determined in Equation 1.

$$P(w_t \mid w_{<t}, \text{context}, \text{emotion}) \tag{1}$$

Where w_t is the proceeding word in the dialogue. We elected to fine tune an existing model rather than training a new Seq2Seq model from scratch. [6] explains many key benefits for transfer learning. The main benefits of transfer learning that pertained to our group were: 1) Transfer learning has proved to be successful given the constraint of limited data (in comparison to the original training dataset). 2) Limited compute resources (model was trained using an NVIDIA A100 GPU on Google Colab pro tier for approximately 4 hours).

3.2. Response Generation Dataset

While text messages use slightly different tones and vocabularies, we tried to find any labeled dataset that contained labeled conversational text. The training data used to fine tune the model was publicly available via Huggingface [2]. The dataset that we selected contained over 15,000 labeled conversations between two individuals, these conversations contained multiple lines of dialogue, and thus, needed to be broken up into multiple input and response pairs. The labels provided an emotion for each line of dialogue in the conversation. These matched our labels for facial reaction (no emotion, anger, disgust, fear, happiness, sadness, and surprise). The structure of the dataset was changed in order to fit a more traditional sample-label value. Conversations were broken into input and response pairs, with a special token indicating the emotion of the response and another token indicating the context of the conversation. An example of how we transformed the original dataset to be fit for training is shown below.

Dataset Example

Original dialogue: `["How is your day going?", "My day is going great!", "Really? I thought you said earlier you were having a bad day."]`
Original emotion: `[no emotion, happy, surprise]`
Transformed:
```
{"input": emotion: Happy | context: | "How is your
day going?", "response": "My day is going great!"}
{"input": emotion: Surprise | context: | "My day is
going great!", "response": "Really? I thought you
said earlier you were having a bad day."}
```

3.3. Model Selection and Hyperparameter Tuning

We extensively researched seq2seq transformer models for this task. Our group selected Google's T5 model, introduced by [5] This model was originally trained for a wide variety of seq2seq tasks (translation, summarization, sentiment), which allows this model to generalize to new tasks relatively well via fine tuning. Additionally, a small version of T5 (T5-small) offers flexibility when it comes to running the model on a smaller device. T5-small has 60.5 million parameters and was originally trained over a wide variety of seq2seq tasks including translation and summarization.

Some of the concerns that we had going into training included overfitting occurring very early into the training stage (within the first 3 epochs) due to the relatively small amount of training data and the computational cost of a grid search. This rapid overfitting was observed with a relatively large learning rate of 1e-3, we restarted the fine-tuning with a smaller learning rate.

Rather than solely querying the existing model for a generated response, we fine-tuned and validated T5-small for conversation-style texts based on emotion using the data mentioned above (~70,000 training input/response samples, 7069 input/response samples, and 6740 test input/response samples). Additionally, our group added in special tokens to represent context, the emotion of the response and the response itself allowing learning a model to define a relationship between the three. This style of emotion-based responses was a novel approach to fine-tuning the T5-small model. We limited the tokenized length to 50 to both speed up the training process and ensure that shorter responses were generated (to match the style of texting). This presented a significant opportunity for future research that we discuss at the conclusion of this report. During training, we elected to use the tokenizer from the original T5-small and focused a majority of time on fine tuning.

We elected to fine-tune the model using the model's original parameters. This allowed us to efficiently use the training process to tune the parameters to learn conversational-style text. At the start of training, a learning rate of 5e-4 was set. We expected that overfitting would occur quickly given this learning rate, so we decreased this value to 1e-4 following the 5th epoch. We scheduled a cosine learning rate [3]. All hyperparameters can be found in Table 3. For text prediction, we used a greedy approach, selecting the next word with the highest probability. The loss function was cross-entropy.

Hyperparameter	**Value**
Learning Rate	5e-4 (ep. 0–5), 1e-4 (ep. 6–20)
Training Epochs	25
Batch Size (per device)	16
Weight Decay	0.01
Early Stopping	TRUE (pat. = 3)

Table 3. Model Hyperparameters

3.4. Response Generation Evaluation

Results were evaluated both qualitatively and quantitatively.

The loss and perplexity of the model were used as quantitative evaluation metrics. The cross-entropy losses for the training and validation data were 0.63 and 0.83 respectively. The model exhibited some evidence of overfitting around the 20th epoch, and thus the training was stopped at epoch 20. Figure 8 shows that the perplexity of the training and validation data was 1.80 and 2.3 respectively.

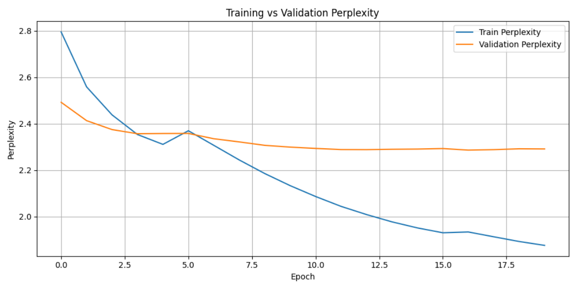

Figure 8. Response Generation Model Perplexity

Qualitatively, we found that the best way of evaluating the model was selecting a random "text" from the testing data set, passing through all the available emotions and evaluating how well the model captured the emotion in the response. The model performed well in this regard when compared to the base T5-small model (which just returned the same text). It's worth noting that the model performed sub-optimally when qualitatively compared to GPT 4o[1]. This was expected, as GPT 4o is over 3,000 times larger than our fine-tuned model. An example of query responses against T5-small (base, 60.5M params), T5-small (our fine-tuned model, 60.5M params), GPT-4o, (200+B params) is shown below.

Example Response Comparison

Input: `emotion: Happy | context: "Tell me how you're feeling"`

T5 Base: Tell me how you're feeling.

T5 (Fine-Tuned): I'm feeling pretty good.

GPT-4o: I'm feeling really great, actually! Thanks for asking — how about you?

4. Other Experiments and Results for Facial Emotion Recognition

4.1. Overview

Additionally, We experimented with three different models: Softmax Regression, Two-Layer MLP, and another version of CNN, we will call it "second CNN variant" in this section. We measured success using accuracy on training, validation, and test datasets.

The Softmax Regression model was simple, using a fully-connected layer with a ReLU activation. It had a very short runtime of about 0.0001 seconds per epoch, making it efficient. We trained it for 50 epochs using a batch size of 64, learning rate of 0.0025, regularization rate of 0.01, and momentum of 0.9.

However, it only achieved 34.69% accuracy on the training set and 32.86% on the test set, as shown in Figure 9 Its simplicity likely limited its ability to capture complex features in facial images, causing underfitting.

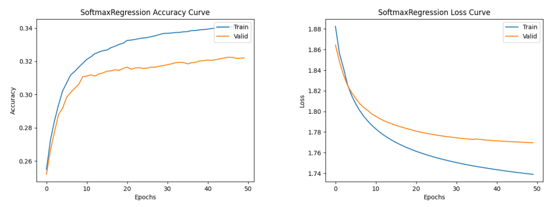

Figure 9. Softmax Regression Results

The Two-Layer MLP added one more fully-connected layer and used a Sigmoid activation be-

[1]GPT-4o was prompted: "Respond to this text message given emotion: [emotion]. [text-message]"

tween layers. It ran quickly (0.001 seconds per epoch) and performed slightly better than the Softmax model. It was trained for 10 epochs with a batch size of 32, learning rate of 0.04, regularization rate of 0.01, and momentum of 0.9.

Figure 10 shows that the model achieved 32.43% training accuracy and 34.16% test accuracy. This modest improvement suggested the extra layer helped slightly, but the use of Sigmoid activations may have caused vanishing gradients and slow learning.

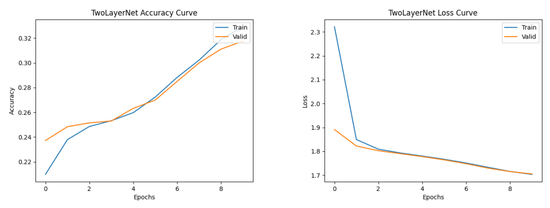

Figure 10. Two-Layer MLP Results

The second CNN variant was more complex than the Softmax Regression and Two-Layer MLP models. It has a different configuration than the CNN model discussed in the Approach section that renders a better result. It had two convolutional layers with ReLU activations and batch normalization, followed by fully connected layers and dropout for regularization. We trained it for 200 epochs using a batch size of 400, learning rate of 0.05, regularization rate of 0.006, momentum of 0.8, with a warmup phase and learning rate decay at steps 120 and 160. We used A100 GPU to significantly reduce the runtime to 0.0076 seconds per epoch.

Figure 11 shows that the second CNN variant achieved 63.83% accuracy on the training data and 54.97% on the test data. It outperformed the other two models by a wide margin, likely due to its ability to extract spatial features from images.

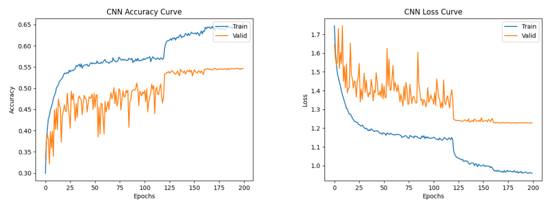

Figure 11. Other CNN Results

However, the second CNN variant still fell slightly short of our best CNN model discussed in the Approach section. Its relatively high training accuracy but lower test accuracy suggested mild overfitting. We used the second CNN variant model to predict emotions for three new angry images in Figure 12, and the results are *angry*, *sad*, and *neutral*, respectively.

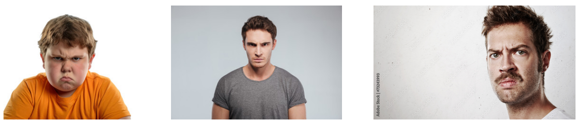

Figure 12. Sample images labeled as *angry* by humans, collected from online sources.

Comparing these experiments, we found that deeper and image-aware architectures are far more effective for facial recognition tasks. Simpler models failed to learn meaningful patterns due to their limited capacity or poor activation choices. This experiment confirmed that model choice has a major impact on performance, even when trained on the same data.

5. Future Research

There remain numerous avenues for advancing research in the domain of language and response modeling. One promising direction involves extending the context window during seq2seq model training - not solely to accommodate longer user messages, but also to incorporate conversational tone and style by integrating preceding dialogue. This expanded context would enable the model to better infer the relational dynamics between the sender and recipient (e.g., professional versus personal interactions), thereby improving the relevance and appropriateness of its responses.

Additionally, enhancing the emotional responsiveness of compact language models - particularly in comparison to larger models such as GPT-4o could be achieved by obtaining a more balanced and diverse training dataset. The dataset utilized in this study demonstrated a pronounced class imbalance, with neutral responses disproportionately represented. This skew likely introduced a systematic bias toward neutral outputs, undermining the model's capacity to generate emotionally congruent responses. Addressing such imbalances would be critical in enabling smaller models to more accurately capture and reflect the emotional nuances expressed by users.

Team Contributions

A summary of team member contributions is provided in Table 4.

Student Name	**Contributed Aspects**	**Details**
Garrard, Jeremy	Facial Emotion Recognition Implementation	Created and tuned CNN model for emotion recognition.
O'Donnell, Liam	Response Generation	Data transformation and seq2seq fine tuning for response generation. Conducted qualitative and quantitative experiments for model
Taskovski, Kelly	Facial Emotion Recognition Implementation	Experimented varying deep learning models for facial recognition and compared the results.

Table 4. Contributions of team members.

References

[1] 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. Challenges in representation learning: A report on three machine learning contests. *arXiv preprint arXiv:1307.0414*, 2013.

[2] Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. Dailydialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of The 8th International Joint Conference on Natural Language Processing (IJCNLP 2017)*, 2017.

[3] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2017.

[4] Albert Mehrabian. *Nonverbal Communication.* Aldine-Atherton, Chicago, IL, 1972.

[5] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023.

[6] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning, 2020.