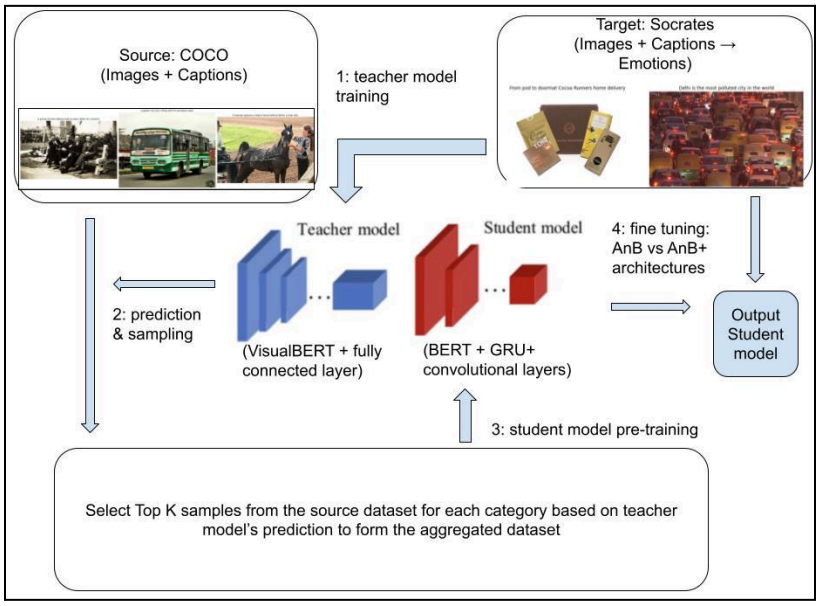
Emotionally Aware Image Caption Generation

[Github Link]

Methodology:

Inspired by [Yalnize et Al], we employ a student-teacher architecture for the transfer learning process. We first train a teacher model on the target dataset, make predictions on the source using the teacher model, resulting in a new dataset. We then pretrain a student model on the new dataset, then finetune it for better predictions. The main reason for adopting this approach is that our target dataset is a lot smaller than the source, so transferability is perfect for our task. The image below summarizes our training approach.



Step 1: Teacher Model Training

Teacher model architecture: The teacher model is trained on the Socratis dataset, and consists of a VisualBERT layer (a variant of BERT that interprets both textual and visual inputs) followed by a fully connected layer. Since VisualBERT takes in image embeddings rather than raw embeddings, we use a pre-trained Fast R-CNN to extract the needed image embeddings and variables. We use a fully connected layer after the VisualBERT layer to map the high-dimensional pooled output from VisualBERT to a lower-dimensional space corresponding to the number of emotions. The architecture of the teacher model is shown to the right.

Teacher Model Training: We use Binary cross-entropy with logit loss (BCEWithLogitsLoss) as our loss function because our class is a multi-label classification task: labels are not mutually exclusive.

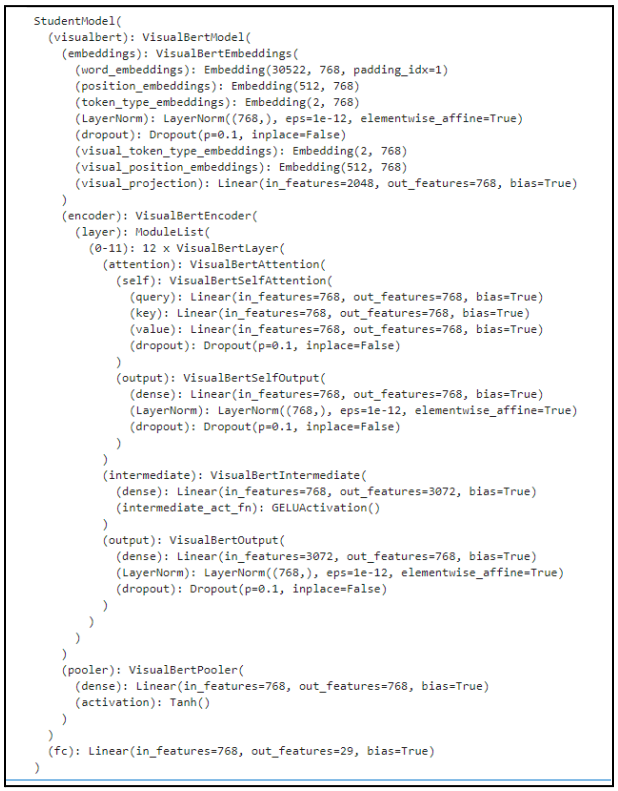
```
TeacherModel(
  (visualbert): VisualBertModel(
    (embeddings): VisualBertEmbeddings(
      (word_embeddings): Embedding(30522, 768, padding_idx=1)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
      (visual_token_type_embeddings): Embedding(2, 768)
      (visual_position_embeddings): Embedding(512, 768)
      (visual_projection): Linear(in_features=2048, out_features=768, bias=True)
    )
    (encoder): VisualBertEncoder(
      (layer): ModuleList(
        (0-11): 12 x VisualBertLayer(
          (attention): VisualBertAttention(
            (self): VisualBertSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): VisualBertSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): VisualBertIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): VisualBertOutput(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
    (pooler): VisualBertPooler(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (activation): Tanh()
    )
  )
  (fc): Linear(in_features=768, out_features=29, bias=True)
)
```

Step 2:COCO prediction and sampling

The teacher model in Step 1 was then used to make emotion predictions on each image-caption pair in COCO. We obtained probabilities for each of the 29 emotion classes for each image-caption pair in COCO. The value of **P** we chose, which represents the P highest scores for each image that would be retained, was 4. We decided on 4 as it is the approximate average number of emotions per image in Socratis. Because our value for **P** is relatively small, we don't assign a threshold value for K and instead keep all the images assigned to each emotion class.

Step 3: Student model Pre-training

Originally, we attempted to use the entire COCO dataset (~100k points) that has been labeled by the teacher model but this would have been too computationally expensive per epoch. Ultimately, we decided on sampling 10% of the COCO dataset, which greatly reduced the training time. The architecture of the student model is shown in the image on the right: it is the same architecture as the teacher model, a VisualBERT layer followed by a fully connected layer. We choose the same architecture because we want to ensure there is a smooth transferring learning process from the teacher to the student. We had previously chosen an overly simple student model that resulted in incompatible layers for transfer. Because of time constraints, we only pre-train for one epoch.



```
StudentModel(
  (visualbert): VisualBertModel(
    (embeddings): VisualBertEmbeddings(
      (word_embeddings): Embedding(30522, 768, padding_idx=1)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
      (visual_token_type_embeddings): Embedding(2, 768)
      (visual_position_embeddings): Embedding(512, 768)
      (visual_projection): Linear(in_features=2048, out_features=768, bias=True)
    )
    (encoder): VisualBertEncoder(
      (layer): ModuleList(
        (0-11): 12 x VisualBertLayer(
          (attention): VisualBertAttention(
            (self): VisualBertSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): VisualBertSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): VisualBertIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): VisualBertOutput(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
    (pooler): VisualBertPooler(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (activation): Tanh()
    )
  )
  (fc): Linear(in_features=768, out_features=29, bias=True)
)
```

Step 4: Fine Tuning:

Given our student model (A) from Step 3, and the teacher model (B) from Step 1, we will experiment with 2 fine tuning approaches based on Yosinski et al. In both cases, we initialize the parameters of the teacher model (B) with the pre-trained weights of the student model (A). The two techniques are described below:

- AnB: fine tuning with all the layers (except the last fc layer) frozen
- AnB+: fine tuning but not freezing the layers

After fine tuning, we evaluate on the Socratis Dataset to see if transferring information from COCO aids performance.

Step 5: Metric calculation and evaluation

Given our task is a multi-label classification task (labels are not mutually exclusive), we use "Top-K" metrics to evaluate performance. These metrics are listed below:

- Top-k Accuracy: We consider the top-1, top-2 and top-3 accuracies. Examining more than 4 could have led us to overestimate the capabilities of the models as there are less penalties. This metric is

calculated as follows: we consider the top-k predicted labels for each sample and check if the ground truth label is among these top predictions. This metric then measures the proportion of samples where **at least one** of the ground truth labels matches the top-k predictions.

- Top-3 Recall: This metric measures how well the model captures the relevant labels. For each sample, we look at the ground truth labels and determine what fraction of these are captured among the model's top-3 predictions. We then divide the number of correct predictions that appear in the top-3 labels by the total number of relevant labels.

Results & Analysis

Training teacher model: The training graph for the Socratis teacher model demonstrates effective convergence, with a significant decrease in loss occurring within the initial epochs and stabilizing at a low level around epoch 10. This indicates that the model has rapidly learned the patterns necessary for its emotion classification tasks. The stable loss beyond the initial learning phase suggests that the model is well-optimized.

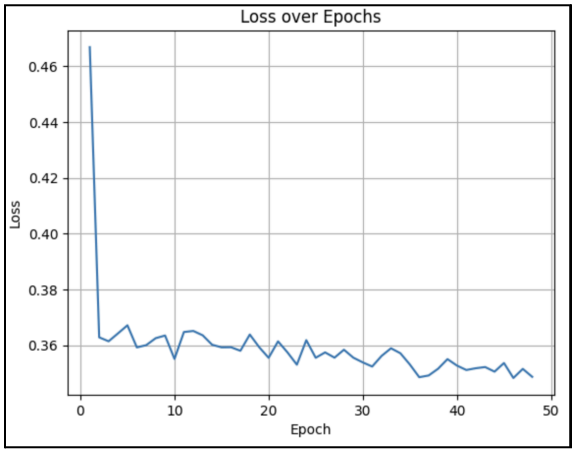


Figure 1: Socratis teacher model loss over epochs

Training student model: As explained previously, due time and compute resource limitations, we were only able to train one epoch of our student model.The graph displaying the training loss of the student model over each batch shows significant fluctuation, with loss values ranging broadly between approximately 0.2 and 0.9. This is however expected as the loss tends to fluctuate wildly over one epoch.

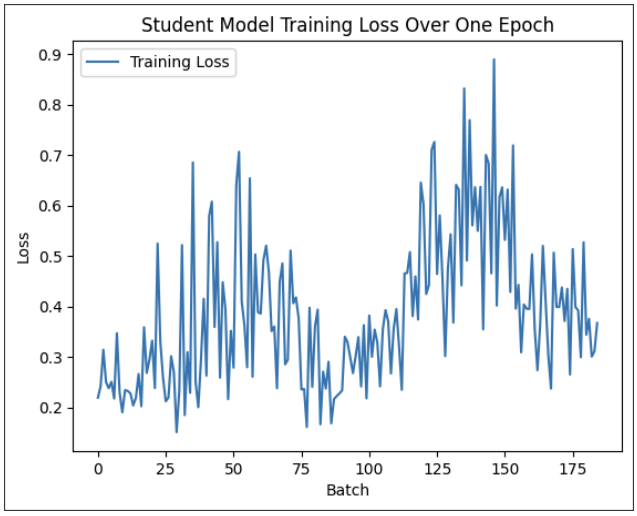


Figure 2: student model training loss over one epoch

Fine Tuning: We explored two fine-tuning techniques, AnB and AnB+, to adapt a teacher model initialized with the student model's pre-trained weights.

The performance graph over 10 epochs illustrates that, AnB, with all layers frozen except the final one, exhibits a rapid decrease in loss, leveling off quickly to a lower steady state. This indicates a robust adaptation to the dataset, capturing complex data patterns efficiently. Conversely, AnB+, which allows all layers to be fine-tuned, shows a slower and more conservative loss reduction, suggesting limited learning from the data despite unlimited flexibility in fine-tuning. The results from examining the loss alone, could suggest that the increased flexibility from unfreezing layers makes it harder for the model to converge.

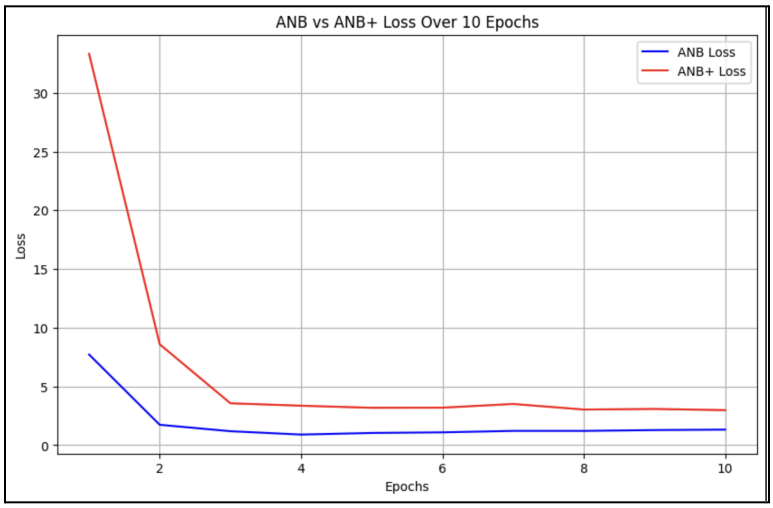


Figure 3: AnB and AnB+ loss comparison over 10 epochs

The comparison of the metrics between both architectures however, shows opposite results than the loss graph. The AnB+ architecture outperforms the AnB architecture across all metrics. This suggests that despite the difficulty with convergence, the AnB+ model is able to make better predictions, meaning that the transfer of knowledge aids with predictive capabilities. The low recall values however, suggest that both models fail to capture true positive labels. Perhaps training over more epochs could have made the models better, but both architectures seem to flatline around the 3rd epoch.

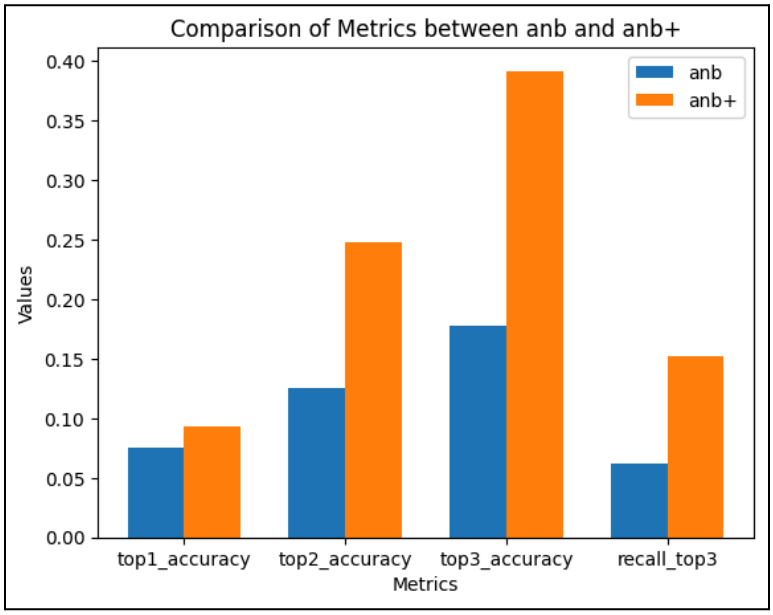


Figure 4: Comparison of Metrics between AnB and AnB+

Conclusions:
Although the methodology is drastically different from our Milestone 1 proposal, we have been able to successfully test the capabilities of emotionally aware caption generation using transfer learning. Further, we conducted additional experiments with AnB and AnB+ techniques, with the AnB+ architecture showing more promising results.

Some challenges we faced included:
1. Limited Training Epochs: Due to impractical training times, we were unable to train for many epochs.
2. Dataset Limitations: We initially planned to use the entire COCO Dataset but had to settle for a sample due to our limited computing resources. This limitation potentially affected the diversity and volume of training data, impacting the model's performance and generalizability for the AnB and AnB+ architectures.

For future directions, several improvements are suggested:
1. Upgrading Computational Resources: Access to more powerful computational resources would allow us to train our models for more epochs as well as use more of the COCO Dataset.
2. Expanding Model Testing: We can explore testing with a wider set of metrics as well as more datasets.

Workflow:
Teacher model, Fine-tuning: Tomisin
Student model: Christine
Report: Christine, Tomisin, Minjoo