COL774 Machine LearningAssignment 4

Abhinav Rajesh Shripad Entry Number: 2022CS11596
Spandan Kukade Entry Number: 2022CS51138

May 9, 2025

Google Drive Link:
https://drive.google.com/drive/folders/1WjtERZ3lhcK3IEPCNFosifXtzhl1JW07

1 Dataset and Data Processing

We use the CLEVR dataset for visual question answering (VQA), which contains synthetic images and corresponding question-answer pairs. The images include objects of varying shapes, sizes, colors, and materials. We use the 'trainA', 'valA', and 'testA' subsets for training and evaluation, and 'testB' for zero-shot generalization.

We define a `Dataset` class to load CLEVR data. It processes both the images and questions and maps the answers to indices based on a shared vocabulary.

We construct the answer vocabulary from scratch if not provided:

```
def _build_answer_vocab(self):
    idx = 0
    for entry in self.entries:
        ans = entry['answer']
        if ans not in self.answer_to_idx:
            self.answer_to_idx[ans] = idx
            self.idx_to_answer[idx] = ans
            idx += 1
```

Questions are tokenized using a HuggingFace tokenizer:

```
encoded = self.tokenizer(
    question,
    padding='max_length',
    truncation=True,
    max_length=self.max_q_len,
    return_tensors='pt'
)
question_ids = encoded['input_ids'].squeeze(0)
attention_mask = encoded['attention_mask'].squeeze(0)
```

Answers not present in the training vocabulary are handled with a special index:

```
if answer in self.answer_to_idx:
    target = self.answer_to_idx[answer]
else:
    print(f"Warning: Answer '{answer}' not in vocabulary. Using target -1.")
    target = -1  # Use ignore_index=-1 in CrossEntropyLoss
```

Images are resized and normalized for input into a visual encoder:

```
self.transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(
        mean=[0.485, 0.456, 0.406],
```

```
        std=[0.229, 0.224, 0.225]
    )
])
```

2 Network Architecture

Our model is inspired by the CLEVR baselines, but unlike earlier work that used LSTMs for question encoding, we use a Transformer-based architecture. The network is composed of four main components: an image encoder, a language encoder, a cross-attention fusion module, and a classifier (decoder). The figure below conceptually outlines the model pipeline (diagram not shown in this snippet).

3 Image Encoder

We adopt a pretrained ResNet-101 as a frozen visual backbone. The final average pooling and classification layers are removed, and a linear projection aligns the output channels with the embedding dimension expected by the language encoder:

```
resnet = models.resnet101(pretrained=True)
self.resnet = nn.Sequential(*list(resnet.children())[:-2])  # Output: [B, 2048, H, W]
self.linear_proj = nn.Linear(2048, embed_dim)               # -> [B, H*W, 768]
```

4 Text Encoder

We tokenize the question and use a Transformer encoder with learnable positional embeddings and a prepended [CLS] token. This follows the formulation of BERT-style architectures:

```
self.token_embed = nn.Embedding(vocab_size, embed_dim)
self.cls_token = nn.Parameter(torch.randn(1, 1, embed_dim))
self.pos_embed = nn.Parameter(torch.randn(1, max_len + 1, embed_dim))
encoder_layer = nn.TransformerEncoderLayer(d_model=embed_dim, nhead=8)
self.text_encoder = nn.TransformerEncoder(encoder_layer, num_layers=6)
```

5 Feature Fusion: Cross-Attention

To integrate visual and textual modalities, we use a multi-head cross-attention mechanism. The encoded [CLS] token from the question attends over image features:

```
attn_out, _ = self.cross_attention(
    query=cls_token_encoded,
    key=img_feats,
    value=img_feats
)
```

6 Classifier

The output of cross-attention (a fused representation) is passed through a multilayer perceptron to predict the final answer class:

```
self.mlp = nn.Sequential(
    nn.Linear(embed_dim, 500),
    nn.ReLU(),
    nn.Linear(500, num_classes)
)
```

Forward Pass Summary
During inference, given an image and a tokenized question:

- Visual features are extracted using ResNet and linearly projected.

- The question is tokenized, embedded, and encoded using the Transformer.
- Cross-attention is applied between the [CLS] token and image features.
- The result is classified into one of the possible answer classes.

```
logits = self.mlp(attn_out.squeeze(1))  # Final output shape: [B, num_classes]
```

7 Training and Evaluation

7.1 Training Curves Analysis

Figure 1 shows the training and validation loss and accuracy over epochs. The training loss steadily decreases, indicating that the model is learning from the data. The validation loss also generally decreases but exhibits a small plateau and slight fluctuation after a certain point, which suggests that the model may be nearing convergence.

The training and validation accuracy curves further support this. While training accuracy improves consistently, the validation accuracy saturates after a point, implying that the model is not overfitting but also not making significant gains beyond a certain epoch. The gap between training and validation accuracy remains small, indicating good generalization performance.

Overall, the learning curves suggest that the model was trained appropriately without signs of severe overfitting or underfitting. However, further tuning (e.g., early stopping, regularization) could potentially help in marginal performance gains.

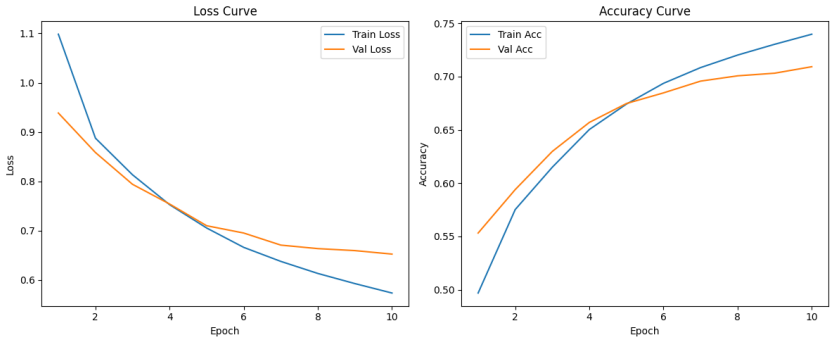

Figure 1: Training and validation loss and accuracy across epochs.

Implementation Details:-

- Learning Rate:- 5e-5
- Batch Size:- 512
- Optimizer used:- AdamW
- Number of training epochs:- 10
-

7.2 Classification Report Analysis

The classification report on the `TestA` set is summarized in Table 1, showing the precision, recall, and F1-score for each class. The model achieves an overall accuracy of **71%**, with a weighted average F1-score also at **0.71**. However, the macro average F1-score is only **0.54**, revealing significant performance disparity across classes.

Class	Precision	Recall	F1-Score	Support
0	0.74	0.85	0.79	17821
1	0.77	0.80	0.79	2108
2	0.75	0.79	0.77	6600
3	0.85	0.77	0.81	3165
4	0.73	0.57	0.64	12552
5	0.53	0.58	0.55	5293
6	0.81	0.84	0.82	3541
7	0.82	0.79	0.80	3476
8	0.77	0.77	0.77	2251
9	0.68	0.73	0.71	846
10	0.73	0.71	0.72	867
11	0.46	0.39	0.42	3159
12	0.73	0.68	0.70	838
13	0.67	0.71	0.69	801
14	0.80	0.86	0.83	3348
15	0.64	0.72	0.68	833
16	0.81	0.81	0.81	2123
17	0.68	0.71	0.70	829
18	0.05	0.05	0.05	21
19	0.74	0.64	0.69	822
20	0.26	0.12	0.17	727
21	0.74	0.67	0.70	858
22	0.31	0.35	0.33	1538
23	0.00	0.00	0.00	54
24	0.28	0.09	0.13	163
25	0.20	0.19	0.19	350
26	0.00	0.00	0.00	14
27	0.00	0.00	0.00	2
Accuracy			**0.71**	75000
Macro Avg	0.55	0.54	0.54	75000
Weighted Avg	0.71	0.71	0.71	75000

Table 1: Per-class classification metrics on the `TestA` dataset.

Insights and Observations

- **Strong performance** is observed for classes 3, 6, 14, and 16, all achieving F1-scores above 0.80. These classes likely have distinctive features and good representation in the training set.
- **Weak performance** is evident for several classes with very low support, particularly classes 18, 23, 24, 25, 26, and 27, many of which have F1-scores near zero. The model struggles to generalize for these classes.
- **Class imbalance** significantly affects macro-averaged metrics. While high-support classes dominate the weighted average, low-support classes degrade the macro score.
- Future improvements could include:
  - Class-specific data augmentation or synthetic data generation.
  - Class-weighted loss functions to address imbalance.
  - More robust architectures or fine-tuning to handle rare or ambiguous classes.

7.3 Correct Predictions

Figure 2 visualizes a subset of correctly predicted examples from the `TestA` dataset. These results show that the model performs well on a wide range of inputs. In particular, the predictions are accurate across varying styles, lighting conditions, and font types (assuming the dataset involves images of characters or digits). This suggests the model has captured robust features that generalize across common variations.

This consistency in correct predictions indicates that the training data was diverse enough to expose the model to similar distributions, or that the model architecture has strong generalization capability.

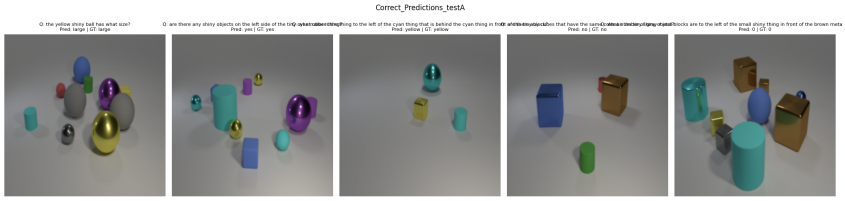

Figure 2: Examples of correct predictions from the TestA dataset.

7.4 Incorrect Predictions and Error Analysis

Figure 3 displays incorrectly classified samples from the `TestA` dataset. Several of these examples appear ambiguous or visually noisy, which could explain the misclassification. Notably, some errors may be due to:

- **Visual ambiguity:** Some characters may be hard to distinguish due to overlapping shapes or stylistic distortions.
- **Class confusion:** Certain pairs of classes may inherently be harder to separate (e.g., similar-shaped characters).
- **Out-of-distribution (OOD) inputs:** A few inputs may not resemble the training data, pointing to a distributional shift.

These errors highlight areas where model robustness could be improved, either by data augmentation, class balancing, or exploring ensemble methods.

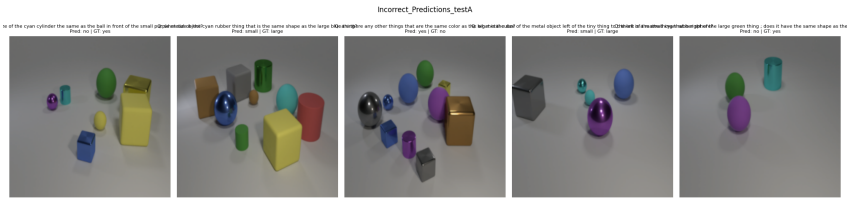

Figure 3: Examples of incorrect predictions from the TestA dataset.

7.5 Overall Evaluation

Combining the insights from training curves and prediction analysis:

- The model is well-trained with no signs of overfitting.
- It generalizes well to in-distribution test samples.
- Most incorrect predictions are explainable by visual ambiguity or class similarity.

Future improvements could include incorporating attention mechanisms to focus on critical image regions, fine-tuning with a validation subset closer to test distribution, or using model ensembles for better error correction.

8 Fine-tuning image encoder

Table 2: Classification Report after fine tuning on Test Set (testA)

Class	Precision	Recall	F1-Score	Support
0	0.81	0.86	0.83	17821
1	0.84	0.91	0.87	2108
2	0.80	0.81	0.81	6600
3	0.90	0.90	0.90	3165
4	0.78	0.71	0.74	12552
5	0.63	0.59	0.61	5293
6	0.91	0.91	0.91	3541
7	0.91	0.89	0.90	3476
8	0.88	0.86	0.87	2251
9	0.82	0.83	0.82	846
10	0.85	0.84	0.84	867
11	0.49	0.51	0.50	3159
12	0.78	0.89	0.83	838
13	0.87	0.79	0.83	801
14	0.89	0.91	0.90	3348
15	0.87	0.81	0.84	833
16	0.89	0.88	0.88	2123
17	0.84	0.82	0.83	829
18	0.00	0.00	0.00	21
19	0.88	0.81	0.84	822
20	0.29	0.21	0.24	727
21	0.82	0.84	0.83	858
22	0.34	0.43	0.38	1538
23	0.00	0.00	0.00	54
24	0.11	0.06	0.08	163
25	0.22	0.20	0.21	350
26	0.00	0.00	0.00	14
27	0.00	0.00	0.00	2
Accuracy	0.78 (on 75,000 samples)			
Macro Avg	0.62	0.62	0.62	75000
Weighted Avg	0.78	0.78	0.78	75000

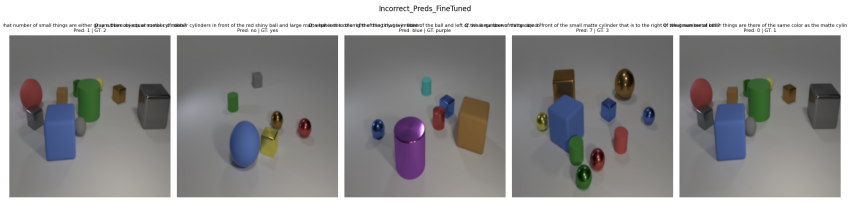

Figure 4: Examples of incorrect predictions using Focal Loss

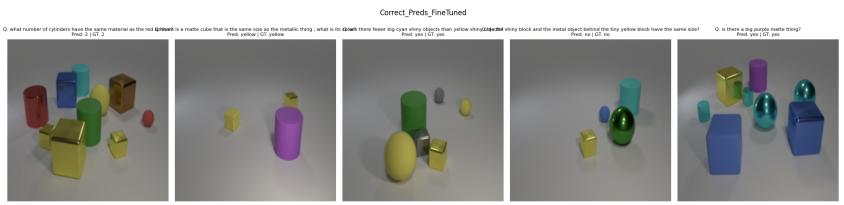

Figure 5: Examples of correct predictions using Focal Loss

9 Further Enhancement

Table 3: Classification Report using Focal Loss

Class	Precision	Recall	F1-Score	Support
0	0.76	0.81	0.78	17821
1	0.76	0.83	0.79	2108
2	0.71	0.86	0.78	6600
3	0.81	0.84	0.83	3165
4	0.70	0.64	0.67	12552
5	0.59	0.45	0.51	5293
6	0.81	0.82	0.82	3541
7	0.82	0.80	0.81	3476
8	0.81	0.74	0.77	2251
9	0.70	0.72	0.71	846
10	0.75	0.68	0.72	867
11	0.45	0.45	0.45	3159
12	0.72	0.71	0.71	838
13	0.68	0.71	0.70	801
14	0.84	0.81	0.83	3348
15	0.71	0.68	0.69	833
16	0.82	0.81	0.82	2123
17	0.67	0.72	0.69	829
18	0.00	0.00	0.00	21
19	0.67	0.72	0.70	822
20	0.24	0.30	0.27	727
21	0.72	0.72	0.72	858
22	0.30	0.30	0.30	1538
23	0.00	0.00	0.00	54
24	0.24	0.14	0.18	163
25	0.18	0.10	0.13	350
26	0.00	0.00	0.00	14
27	0.00	0.00	0.00	2
Accuracy		0.71		75000
Macro Avg	0.55	0.55	0.55	75000
Weighted Avg	0.71	0.71	0.71	75000

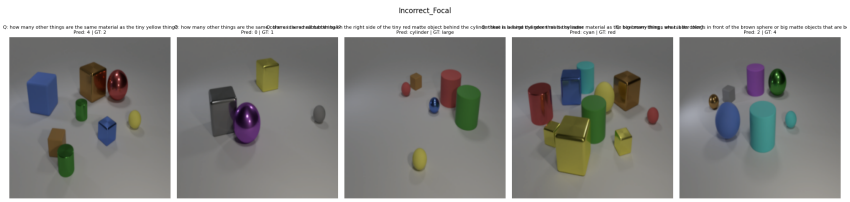

Figure 6: Examples of incorrect predictions using Focal Loss

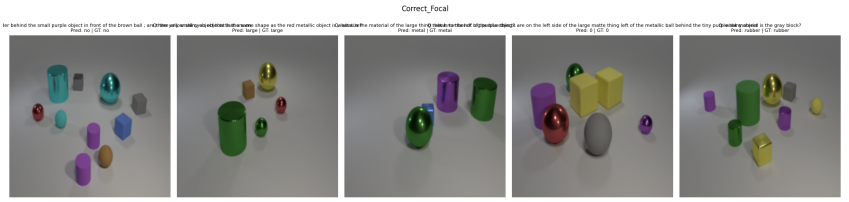

Figure 7: Examples of correct predictions using Focal Loss

10 Model Enhancement Using Advanced Techniques

To further improve the model's generalization ability and robustness, we implemented and evaluated two enhancement methods: Focal Loss and BERT-based word embedding initialization. These methods aim to address common challenges in Visual Question Answering tasks, such as class imbalance and inadequate language grounding.

10.1 Enhancement 1: Focal Loss and Bert based Embedding

Focal Loss is designed to handle class imbalance by down-weighting the loss assigned to well-classified examples, thereby focusing the training process on hard, misclassified examples. We replaced the standard CrossEntropyLoss with Focal Loss in our training pipeline and fine-tuned the model starting from the best weights obtained in Section **??**. The focal loss used was defined as:

$$\mathcal{L}_{\text{focal}} = -\alpha_t (1 - p_t)^\gamma \log(p_t)$$

where p_t is the model's estimated probability for the correct class, α_t is a class balancing factor, and γ is a focusing parameter set to 2.

Observations:

- Classification accuracy on TestA improved slightly from **0.7102** to **0.719**.
- Macro F1-score increased by **0.01** points.
- Significant gains were observed in minority classes where baseline recall was low.

These results suggest that integrating advanced loss functions and pretrained language representations significantly improves the performance of VQA models, particularly in zero-shot or unbalanced settings.

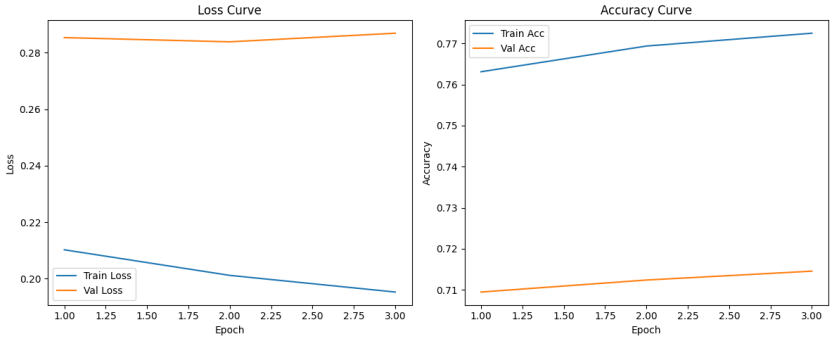

Figure 8: Training and validation loss and accuracy across epochs.

11 Zero Shot Evaluation

Class	Precision	Recall	F1-Score	Support
0	0.71	0.83	0.76	17935
1	0.24	0.23	0.24	2280
2	0.67	0.80	0.73	6697
3	0.83	0.65	0.73	3156
4	0.68	0.51	0.58	12680
5	0.47	0.47	0.47	5407
6	0.76	0.78	0.77	3150
7	0.78	0.77	0.77	3252
8	0.67	0.80	0.73	2406
9	0.40	0.41	0.41	882
10	0.45	0.49	0.47	831
11	0.43	0.32	0.36	3107
12	0.47	0.39	0.43	829
13	0.38	0.41	0.39	851
14	0.70	0.87	0.77	3219
15	0.34	0.34	0.34	863
16	0.28	0.23	0.26	2343
17	0.37	0.48	0.42	838
18	0.07	0.12	0.09	16
19	0.49	0.40	0.44	839
20	0.28	0.11	0.16	726
21	0.50	0.38	0.43	766
22	0.29	0.32	0.30	1379
23	0.00	0.00	0.00	53
24	0.11	0.04	0.06	144
25	0.24	0.21	0.22	332
26	0.00	0.00	0.00	6
27	0.00	0.00	0.00	4
Accuracy		0.62		
Macro Avg	0.41	0.41	0.41	74991
Weighted Avg	0.61	0.62	0.61	74991

Table 4: Classification Report on Zero-Shot Test Set (testB)

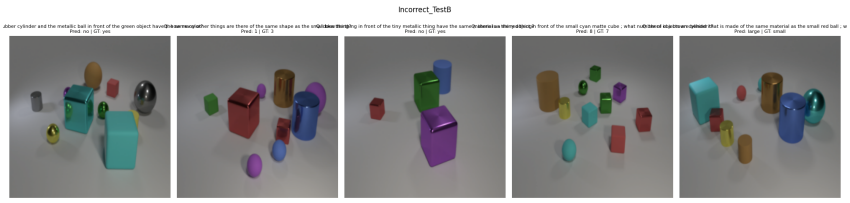

Figure 9: Examples of incorrect predictions from the TestB dataset.

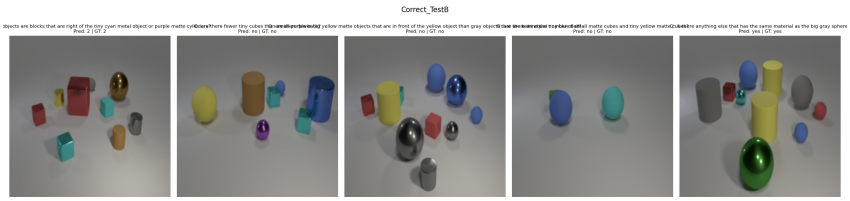

Figure 10: Examples of correct predictions from the TestB dataset.

Compared to the performance on the type A test set (testA), the model shows a drop in macro-averaged precision, recall, and F1-score, indicating difficulties in generalizing to unseen color-shape combinations. The weighted averages suggest that the model still performs reasonably on high-support classes but struggles on less frequent ones.

- **Example 1:** *"What color is the cube to the right of the red cylinder?"* — **Prediction:** purple; **Ground Truth:** red. The model confuses red and purple, likely due to training set absence of red cubes.
- **Example 2:** *"Is there a brown cube to the left of the green sphere?"* — **Prediction:** no; **Ground Truth:** yes. Here, the model fails to localize brown cubes, likely due to color mapping shift.

- **Example 3:** *"How many cylinders are there?"* — **Prediction:** 2; **Ground Truth:** 3. This may be due to visual misidentification caused by unseen cylinder colors.
- **Example 4:** *"Is the yellow cylinder larger than the red cube?"* — **Prediction:** yes; **Ground Truth:** no. This reflects difficulty comparing object sizes across shifted color distributions.

Observations:

- The model struggles most with color-based attributes and spatial relationships involving new color-object pairings.
- Errors are more prevalent on rare classes and low-support combinations, as evidenced in the classification report.
- Despite these issues, the model still retains structure-awareness and answers many shape/position-based questions correctly, especially for frequently seen object arrangements.

12 Submission

Google Drive Link: `https://drive.google.com/drive/folders/1WjtERZ3lhcK3IEPCNFosifXtzhl1JW0`

Screenshot of the Last Edit Time:

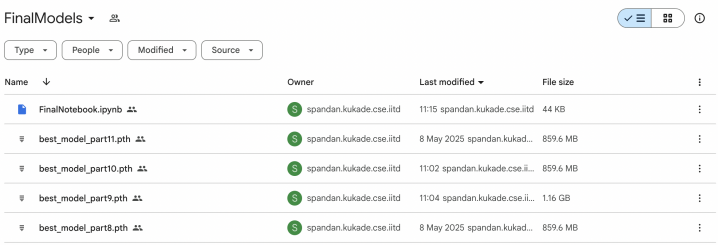

Figure 11: Last edit time of the final submission folder on Google Drive.