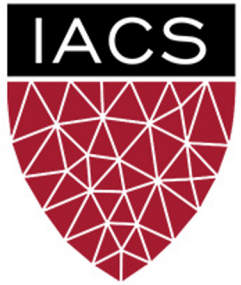
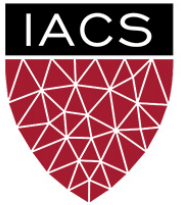


Visual Question Answering

$\alpha\beta normal$ Distriβution

AC295/CS115

Eduardo, Jessica, Rohit & Stuart



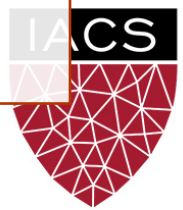
Contents

Overview

VQA Model

Features Implemented

- 20GB Dataset
- TFRecords
- Distillation
- Pruning
- Frontend App



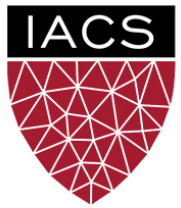
Overview

VQA Model

- Visual Question Answering using transfer learning
- VGG-19 for image feature extraction
- BERT for text feature extraction
- Extracted features combined to determine the response to the questions
- Answers are from the predetermined list of answers

Features

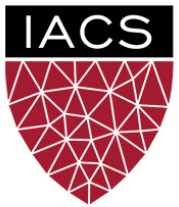
- 20GB Dataset – Requires resizing of the images
- TFRecords – Helpful in case of large datasets requiring significant preprocessing
- Distillation – Reduces model size and complexity leading to speeds up in the inference time
- Pruning – Speeds up the inference time by setting weights below a threshold to zero and thus reducing computation time
- Frontend App – Web interface to interact with the VQA Model



VQA Model

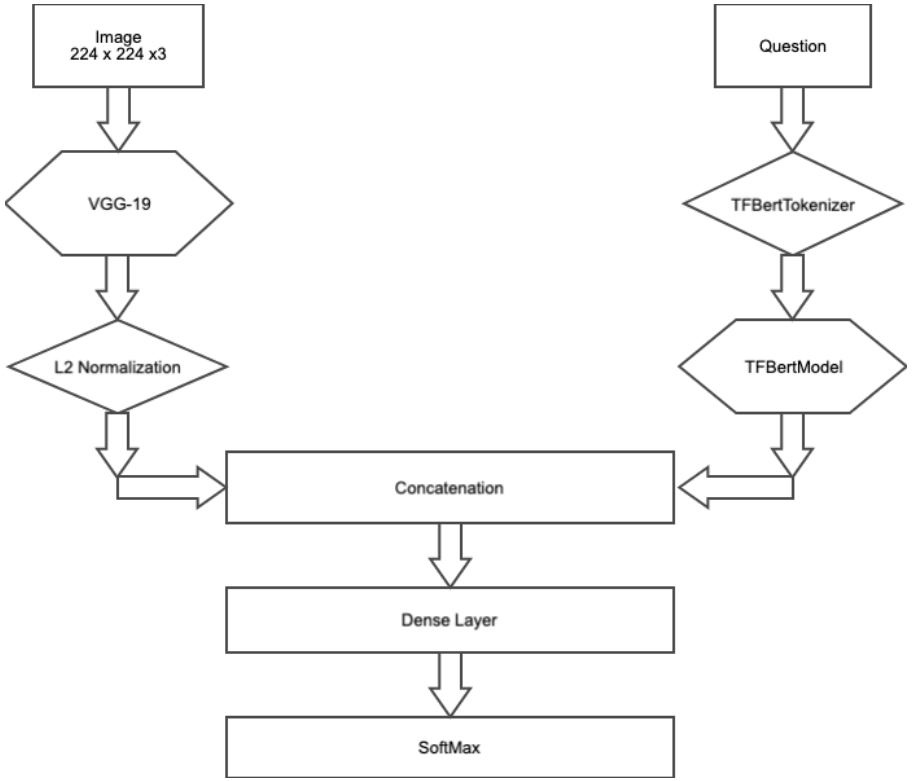
Overview

- For the original model, we use transfer learning by leveraging **VGG19** to extract images feature and **BERT model** to extract the text feature from the questions. We then use concatenation to combine the 2 hidden states and added some dense layer before the output layer.
- The model was trained on the **full (20GB) dataset** in **TFRecords** format
- We achieve accuracy of 49.77%, loss of 0.8791 on validation dataset



VQA Model

Architecture



Parameters & Metrics

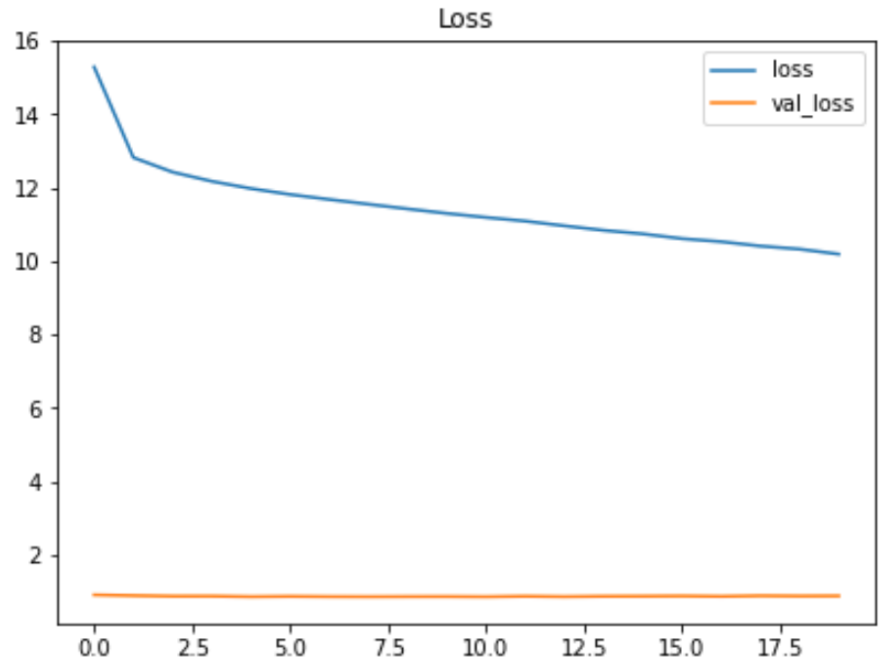
Parameter Type	
Total	155,994,442
Trainable	26,487,818
Non-Trainable	129,506,624

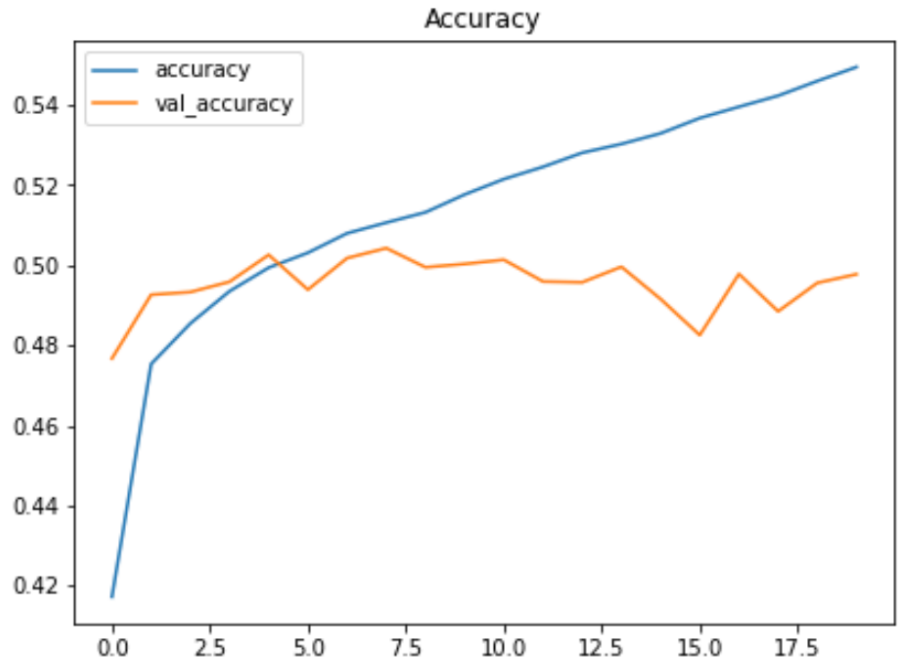
Metric Type	
Validation Cross-Entropy Loss	0.8791
Validation Accuracy	49.77%

IACS

VQA Model

Plot of Model Training and Evaluation Result

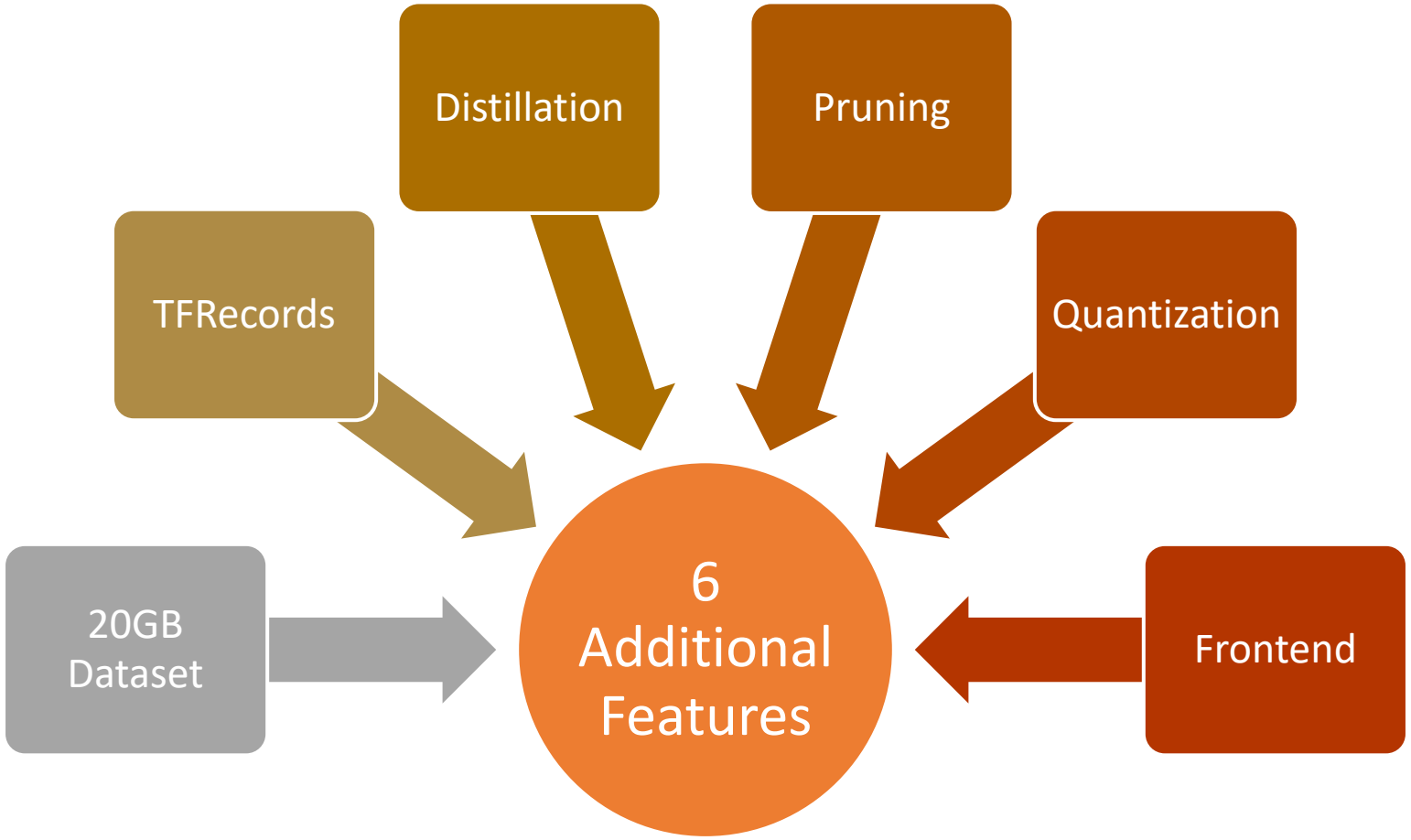





```
3441/3441 [==============================] - 266s 77ms/step - loss: 0.8791 - accuracy: 0.4977
{'loss': 0.8791221380233765, 'accuracy': 0.4977385401725769}
[0.8791221380233765, 0.4977385401725769]
```

IACS

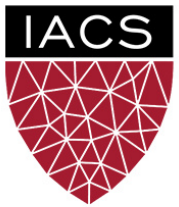
Features Implemented



Feature 1-2: Pipeline with 20GB Dataset & TFRecords

Pipeline Overview

- We first need to resize the images from the full dataset to 224 x 224 x 3
- After preprocessing, we then store the dataset in TFRecords, which reduces the dataset to 8.5GB with sharding
- Further details on the implemenation is in the next slide



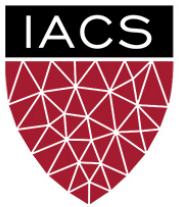
Feature 1-2: Pipeline with 20GB Dataset & TFRecords

Preprocess images

```
# Read and resize the image
image_path = imdir%(
            subset,
            subset,
            anno['annotations'][i]['image_id']
)
img = cv2.imread(image_path)
img = cv2.resize(
            img,
            (IMG_HEIGHT, IMG_WIDTH)
)
img = np.asarray(img)
img = tf.constant(img)
img_raw = tf.io.encode_jpeg(img)
```

TFRecords

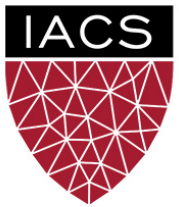
```
# Create tf.train.Example
feature={
            'image_raw': _bytes_feature(img_raw),
            'question' : _bytes_feature(
                        question.encode('utf-8')
            ),
            'input_ids': _bytes_feature(question_input),
            'token_type_ids': _bytes_feature(question_type),
            'attention_mask': _bytes_feature(
                        question_attention
            ),
            'answer': _int64_feature(answer)
}
Features = tf.train.Features(feature=feature)
example = tf.train.Example(features=features)
```



Feature 3: Distillation

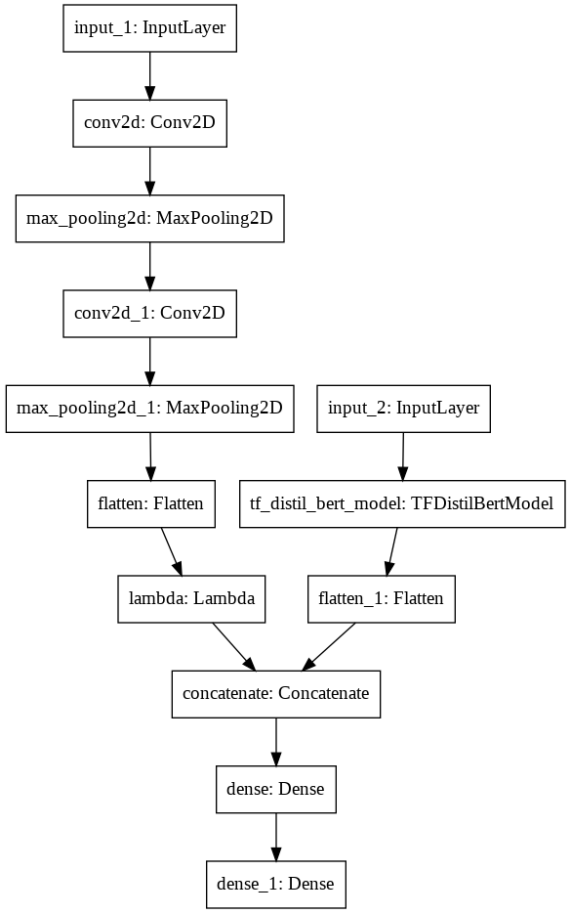
Overview

- Student Model Architecture:
  - In place of VGG19 model, we use 2 blocks of Conv2d & MaxPooling layers (each Cov2d layer using 32 units) to extract image features
  - Use TFDistillBert instead of TFBertModel to extract text features
  - We concatenate the hidden states, and use a dense layer of 32 units before the output layer
- Training the student model with distillation gives us a comparable (slightly higher) validation accuracy of 51.77%.
- Number of trainable parameters is down by ~95%
- Inference time was down by ~60%
- Size of the model was down by ~60%.



Distillation

Architecture



Parameters & Metrics Comparison

Parameters	Original	Distilled
Trainable	26,487,818	801,034

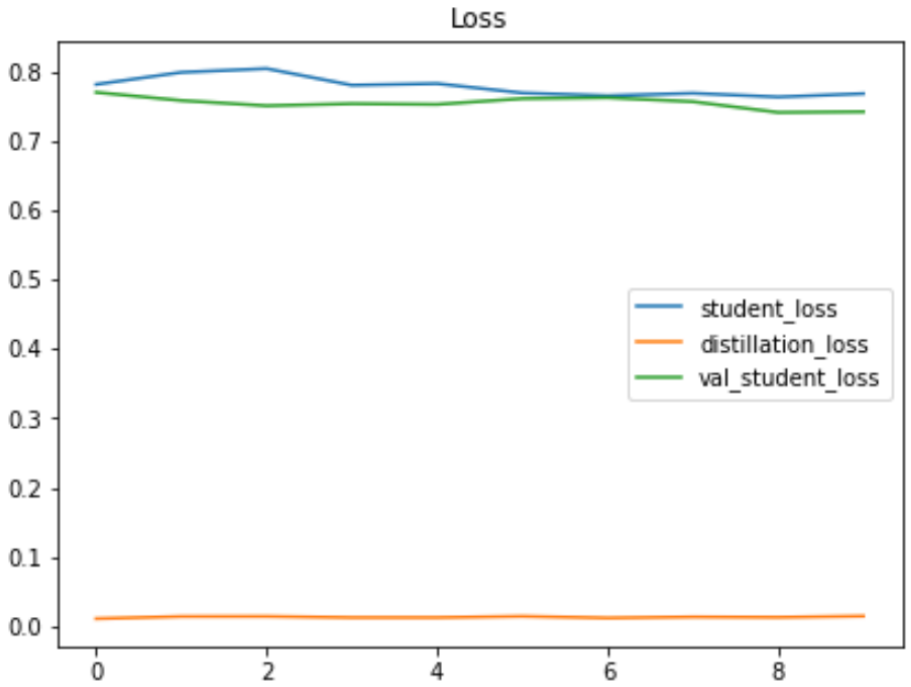
Time	Original	Distilled
Inference	77ms	33ms

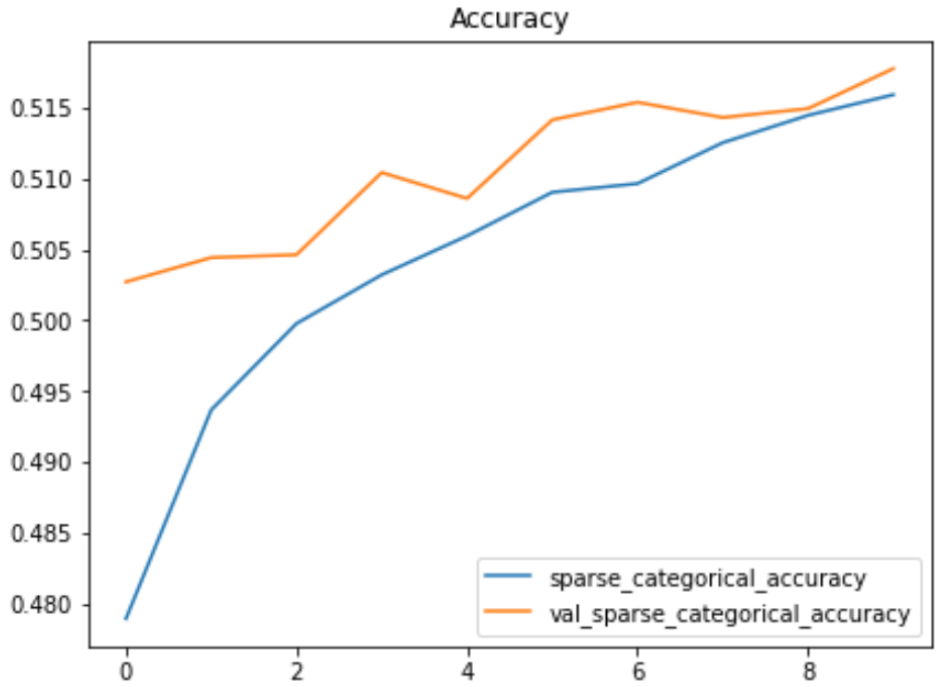
Metric Type	Original	Distilled
Validation Loss	0.8791	0.7419
Validation Accuracy	49.77%	51.77%

Model Size (Weights)	Original	Distilled
Size	610MB	263MB
Compressed Size	565MB	242MB

Distillation

Plot of Model Training and Evaluation Result



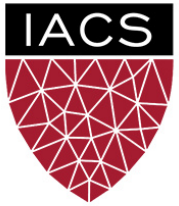



```
3441/3441 [==============================] - 112s 33ms/step - sparse_categorical_accuracy: 0.5177 - student_loss: 0.8298
{'sparse_categorical_accuracy': 0.5177102088928223, 'student_loss': 0.7419548034667969}
```

Feature 4: Pruning

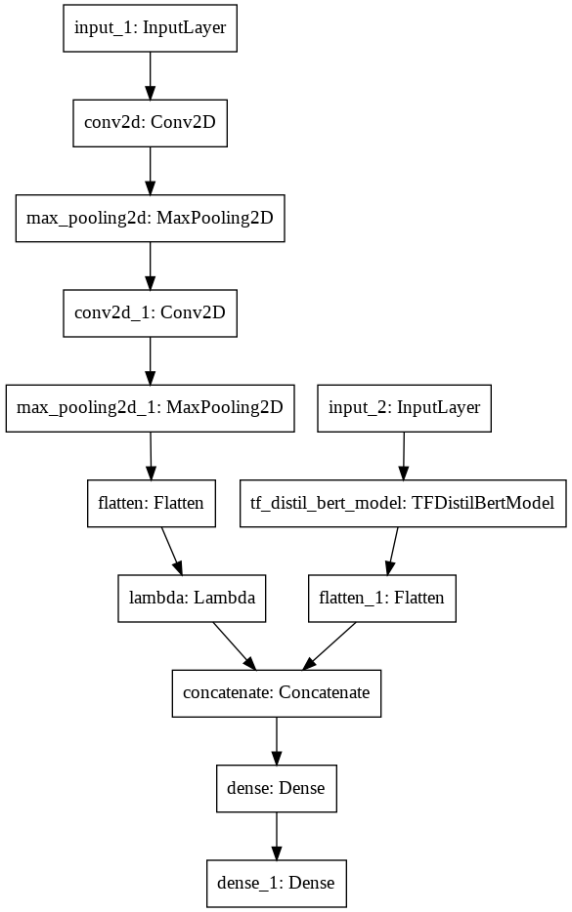
Overview

- For pruning, we started with the distilled version of the model
- The final model (pruned distilled model) has very similar metrics to that of the distilled model.
- Accuracy was down by ~5%
- Observation:
  - Layers from Bert Model were more challenging to be pruned, so for this task, we only did the pruning on the remaining layers. This gives us only a *slight* improvement over the distilled version.
  - Compared to the original model, we still have a big improvement (more than 50% reduction in size, ~95% reduction in parameters). However, reduction was marginal compared to Distilled model.



Distillation + Pruning

Architecture



Parameters & Metrics Comparison

Parameters	Original	Pruned
Trainable	26,487,818	801,034

Time	Original	Pruned
Inference	77ms	32ms

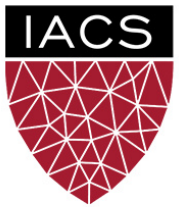
Metric Type	Original	Pruned
Validation Loss	0.8791	1.0833
Validation Accuracy	49.77%	44.13%

Model Size (Weights)	Original	Pruned
Size	610MB	263MB
Compressed Size	565MB	242MB

Feature 5: Quantization

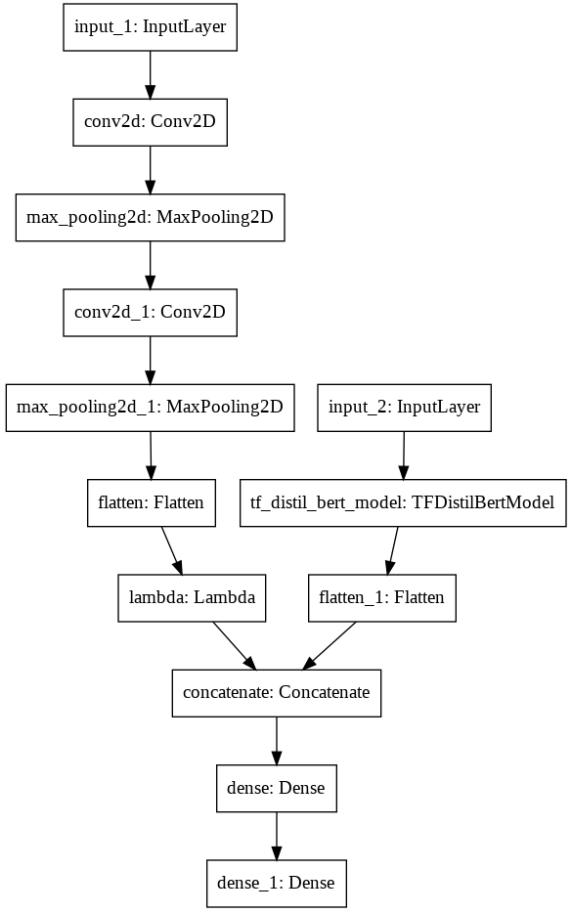
Overview:

- Starting from the pruned distilled model, we then apply quantization as well
- Inference time is slightly higher, mainly because it was run on CPU while the inference time for original model, distilled model, and pruned distilled model were measured when running on GPU.
  - Google Collab crashes when running this model using GPU
- The reduction in model size is now ~80%



Distillation + Pruning + Quantization

Architecture



Parameters & Metrics Comparison

Parameters	Original	Quantized
Trainable	26,487,818	801,034

Time	Original	Quantized
Inference	77ms	83ms on CPU

Metric Type	Original	Quantized
Validation Loss	0.8791	-
Validation Accuracy	49.77%	41.72%

Model Size (Weights)	Original	Quantized
Size	610MB	131MB
Compressed Size	565MB	118MB

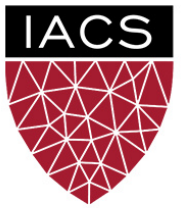
Feature 6: Frontend App

Overview

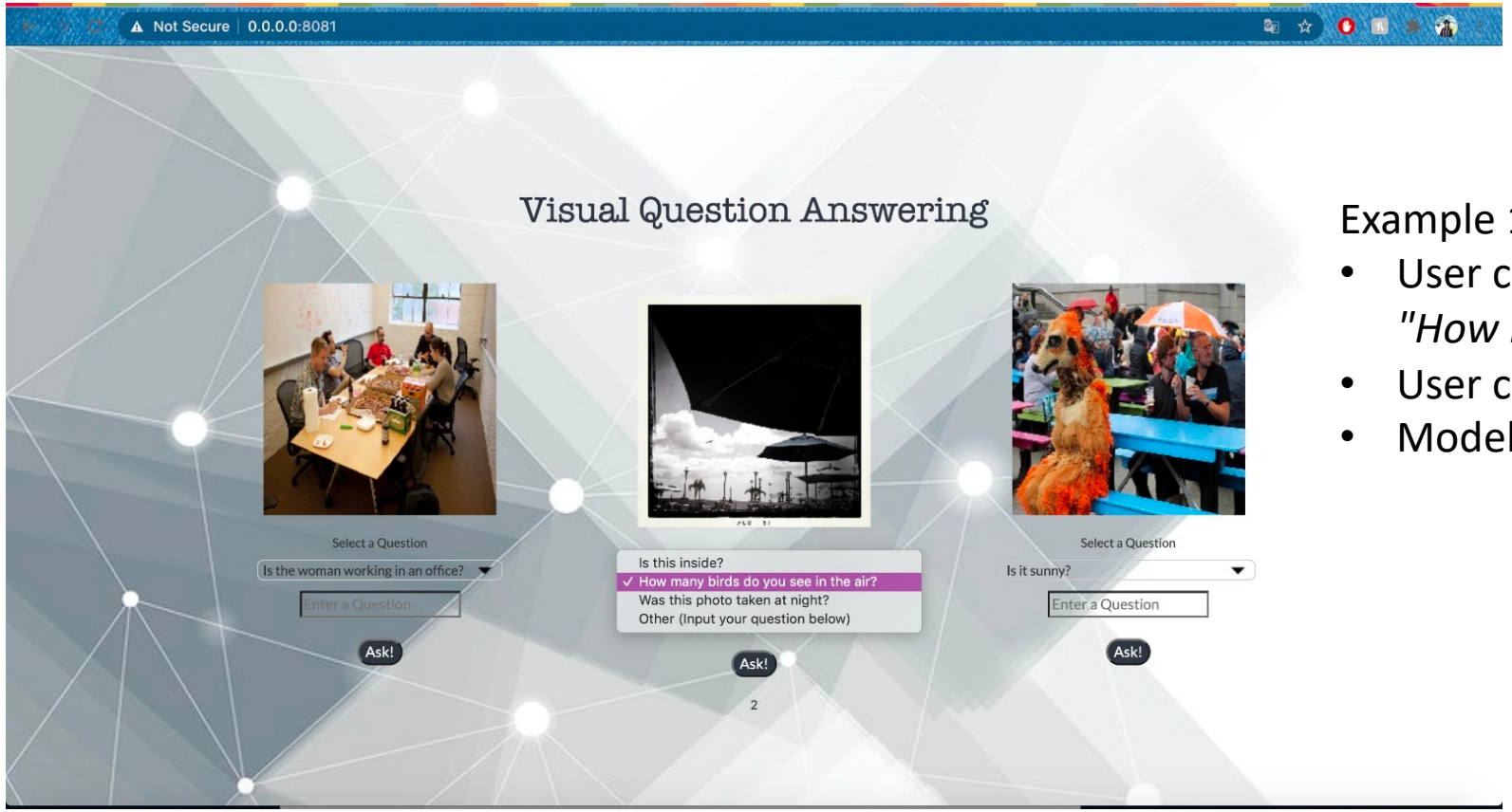
- For the front end, we built a single webpage that display 3 sample images from MSCOCO dataset
- For each image, there is a drop down selections of questions to choose from
- Users can also select the option *'Other'* and then type in their own question in the input box below.
- Once the user click the button *Ask!*, the model will display the predicted answer for the given question

Usage

- We built a docker container for the front end, so user can just build the image and run the container to access the front end UI (code are provided in the README)



Frontend App



Example 1:

- User chose a question from the drop-down options: *"How many birds do you see in the air?"*
- User clicked "*Ask!*" button
- Model predicted the answer: "2"

Frontend App

Example 2:

- User chose from the drop-down selection *"Other (Input your question below)"*
- User typed in question in the input box below: *"How many pizza is there?"*
- User clicked "*Ask!*" button
- Model predicted the answer: "2"

Thanks!