1 Bonus Assignment - CS5720

1.1 Question 1: Question Answering with Transformers

[1]:
```
# Install necessary libraries
!pip install transformers torch --quiet
```

```
                                          363.4/363.4 MB
3.5 MB/s eta 0:00:00
                                          13.8/13.8 MB
52.8 MB/s eta 0:00:00
                                          24.6/24.6 MB
27.4 MB/s eta 0:00:00
                                          883.7/883.7 kB
26.8 MB/s eta 0:00:00
                                         664.8/664.8 MB
968.0 kB/s eta 0:00:00
                                          211.5/211.5 MB
5.1 MB/s eta 0:00:00
                                          56.3/56.3 MB
15.0 MB/s eta 0:00:00
                                          127.9/127.9 MB
8.1 MB/s eta 0:00:00
                                          207.5/207.5 MB
6.6 MB/s eta 0:00:00
                                          21.1/21.1 MB
88.1 MB/s eta 0:00:00
```

[2]:
```
# Step 1: Basic QA Pipeline
from transformers import pipeline

qa_pipeline = pipeline("question-answering")

context = "Charles Babbage is considered the father of the computer."
question = "Who is considered the father of the computer?"

result = qa_pipeline(question=question, context=context)
```

```python
print(result)
```

```
No model was supplied, defaulted to distilbert/distilbert-base-cased-distilled-
squad and revision 564e9b5 (https://huggingface.co/distilbert/distilbert-base-
cased-distilled-squad).
Using a pipeline without specifying a model name and revision in production is
not recommended.
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94:
UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab
(https://huggingface.co/settings/tokens), set it as secret in your Google Colab
and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access
public models or datasets.
  warnings.warn(
```

```
config.json:   0%|          | 0.00/473 [00:00<?, ?B/s]

model.safetensors:   0%|          | 0.00/261M [00:00<?, ?B/s]

tokenizer_config.json:   0%|          | 0.00/49.0 [00:00<?, ?B/s]

vocab.txt:   0%|          | 0.00/213k [00:00<?, ?B/s]

tokenizer.json:   0%|          | 0.00/436k [00:00<?, ?B/s]

Device set to use cpu

{'score': 0.9972344040870667, 'start': 0, 'end': 15, 'answer': 'Charles
Babbage'}
```

[3]:
```python
# Step 2: Using custom pretrained model
qa_pipeline = pipeline("question-answering", model="deepset/
↪roberta-base-squad2")

result = qa_pipeline(question=question, context=context)
print(result)
```

```
config.json:   0%|          | 0.00/571 [00:00<?, ?B/s]

model.safetensors:   0%|          | 0.00/496M [00:00<?, ?B/s]

tokenizer_config.json:   0%|          | 0.00/79.0 [00:00<?, ?B/s]

vocab.json:   0%|          | 0.00/899k [00:00<?, ?B/s]

merges.txt:   0%|          | 0.00/456k [00:00<?, ?B/s]

special_tokens_map.json:   0%|          | 0.00/772 [00:00<?, ?B/s]

Device set to use cpu
```

```
{'score': 0.9830346703529358, 'start': 0, 'end': 15, 'answer': 'Charles
Babbage'}
```

[4]:
```
# Step 3: Custom context and questions
custom_context = "The University of Central Missouri is located in Warrensburg.␣
 ↪It was founded in 1871."

q1 = "Where is the University of Central Missouri located?"
q2 = "When was the university founded?"

print(qa_pipeline(question=q1, context=custom_context))
print(qa_pipeline(question=q2, context=custom_context))
```

```
{'score': 0.9903095960617065, 'start': 49, 'end': 60, 'answer': 'Warrensburg'}
{'score': 0.9294794201850891, 'start': 80, 'end': 84, 'answer': '1871'}
```

1.2 Question 2: Conditional GAN on MNIST

[5]:
```
# Conditional GAN Implementation
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import datasets, transforms
from torch.utils.data import DataLoader
import matplotlib.pyplot as plt

# Hyperparameters
batch_size = 128
z_dim = 100
num_classes = 10
embedding_dim = 50
device = 'cuda' if torch.cuda.is_available() else 'cpu'

# Data
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize([0.5], [0.5])
])

train_loader = DataLoader(
    datasets.MNIST('.', train=True, download=True, transform=transform),
    batch_size=batch_size, shuffle=True
)
```

```
100%|      | 9.91M/9.91M [00:00<00:00, 51.8MB/s]
100%|      | 28.9k/28.9k [00:00<00:00, 1.69MB/s]
100%|      | 1.65M/1.65M [00:00<00:00, 14.4MB/s]
100%|      | 4.54k/4.54k [00:00<00:00, 6.76MB/s]
```

1.2.1 Short Answer

How does a Conditional GAN differ from a vanilla GAN? > Conditional GANs add a condition (like labels) to both generator and discriminator to control the output.

Real-world Application: > Generating faces with specific attributes like smiling, male, or glasses.

What does the discriminator learn in image-to-image GANs? > It learns to distinguish between real and fake pairs. Pairing helps enforce the correct mapping between input and output (e.g., edges to photo).