CS6910: Deep Learning
Assignment 3

Team 7
CS21B007 Arcot Renusree
CS21B036 Janapati Varshita Devi
ED21B042 Naidu Sushant Chandra

May 21, 2025

1 Fine-Tuning BERT for Sentiment Classification

The Bidirectional Encoder Representations from Transformers (BERT) model has achieved state-of-the-art performance on a wide range of NLP tasks. This section documents the fine-tuning process of BERT on a sentiment classification task and evaluates its performance across multiple learning rates: 1e-5, 2e-5, and 3e-5.

1.1 Dataset and Preprocessing

Dataset Overview

- **Training Data:** 5600 samples
- **Test Data:** 1400 samples

Descriptive Statistics for Tweet Length

- **Training Data:**
  - Mean: 12.47 words
  - Standard Deviation: 5.43
  - Range: 1 to 29 words
- **Test Data:**
  - Mean: 12.35 words
  - Standard Deviation: 5.35
  - Range: 2 to 29 words

These statistics indicate that the tweets are relatively short, with most tweets containing between 8 and 16 words. Preprocessing involved standard text cleaning, tokenization, and sequence padding to ensure uniform input length.

1.2 Model Architecture

For this sentiment classification task, we used the `BertForSequenceClassification` model from the Hugging Face `transformers` library. This model extends the pretrained `bert-base-uncased` language model with an additional classification head suitable for downstream tasks such as sentiment analysis.

- **Base Encoder:** `bert-base-uncased`, a BERT model pretrained on lowercased English text from BookCorpus and English Wikipedia.
  - Number of transformer encoder layers: 12
  - Hidden size: 768
  - Number of attention heads: 12
  - Total parameters: approximately 110 million
- **Dropout Layer:** A dropout layer with a probability of 0.1 is applied to the pooled output (i.e., the representation of the [CLS] token).
- **Classification Head:** A linear layer maps the 768-dimensional pooled BERT output to a 2-dimensional vector, representing the binary classes (*Negative* and *Positive*).
- **Softmax Layer:** During evaluation, a softmax function is applied to obtain class probabilities.

1.3 Experimental Setup

- **Model Used:** BERT-base-uncased
- **Task:** Binary sentiment classification (Positive, Negative)
- **Training Data Distribution:**
  - Negative: 5216 samples
  - Positive: 384 samples
- **Test Data Distribution:**
  - Negative: 1291 samples
  - Positive: 109 samples
- **Total Test Set Size:** 1400 samples
- **Hardware:** NVIDIA CUDA-enabled GPU
- **Epochs:** 5
- **Loss Function:** CrossEntropyLoss
- **Optimizer:** **AdamW** (Adam with weight decay)
- **Evaluation Metrics:** Accuracy, Precision, Recall, F1-score, Loss

Before training, the classifier weights (`classifier.weight`, `classifier.bias`) were randomly initialized, and the rest of the model was loaded from the pre-trained BERT-base-uncased checkpoint.

1.4 Training Results

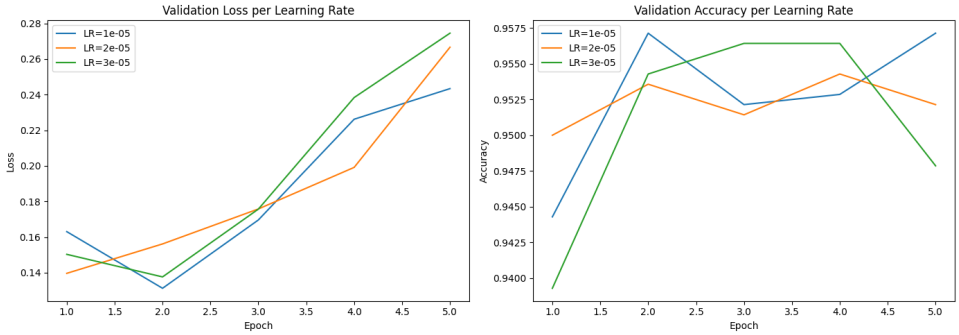


Figure 1.1: Sentiment Analysis Results over different learning rates

Learning Rate: 1e-5

Epoch	Accuracy	Loss	F1 (Pos)	Precision (Pos)	Recall (Pos)
1	94.43%	0.1631	0.47	0.90	0.32
2	95.71%	0.1313	0.68	0.80	0.60
3	95.21%	0.1696	0.67	0.72	0.62
4	95.29%	0.2262	0.60	0.89	0.45
5	95.71%	0.2434	0.69	0.79	0.61

Table 1: Performance Metrics with Learning Rate = 1e-5

Learning Rate: 2e-5

Epoch	Accuracy	Loss	F1 (Pos)	Precision (Pos)	Recall (Pos)
1	95.00%	0.1396	0.60	0.79	0.49
2	95.36%	0.1562	0.61	0.89	0.46
3	95.14%	0.1758	0.66	0.72	0.61
4	95.43%	0.1991	0.70	0.71	0.69
5	95.21%	0.2666	0.63	0.80	0.51

Table 2: Performance Metrics with Learning Rate = 2e-5

Learning Rate: 3e-5

Epoch	Accuracy	Loss	F1 (Pos)	Precision (Pos)	Recall (Pos)
1	93.93%	0.1503	0.44	0.79	0.30
2	95.43%	0.1376	0.62	0.87	0.49
3	95.64%	0.1757	0.64	0.89	0.50
4	95.64%	0.2384	0.68	0.79	0.60
5	94.79%	0.2745	0.68	0.66	0.70

Table 3: Performance Metrics with Learning Rate = 3e-5

1.5 Observations

- Across all learning rates, **accuracy remained above 93%**, showing BERT's robustness for sentiment classification.
- **Learning rate 1e-5** demonstrated consistent F1-scores and the lowest fluctuation in loss.
- The **positive class** (minority) consistently showed lower recall, indicating class imbalance affects the model's ability to identify positive samples.
- With **3e-5**, the model started to **overfit** after epoch 4, seen in rising loss and declining accuracy.
- The best **precision-recall balance** for the positive class was seen with 2e-5 at epoch 4 (F1: 0.70).

1.6 Conclusion

Fine-tuning BERT for sentiment classification achieved high accuracy across all learning rates. The learning rate of **1e-5** yielded the most stable results. To improve performance further, especially on the underrepresented class, class rebalancing strategies or learning rate schedulers should be considered.

2 Task 2: Machine translation using RNN/LSTM model

The dataset is provided in CSV format with two columns: `source` (English sentence) and `target` (corresponding Hindi sentence). The dataset has training, validation and testing subsets.

Preprocessing is performed using the `torchtext` library. Tokenization for English is done using basic whitespace splitting, while for Hindi, we rely on Indic NLP utilities to ensure proper tokenization of Devanagari script. Vocabularies for both languages are built using the training set only, ensuring no data leakage into the test set.

We use pre-trained GloVe embeddings (200-dimensional vectors) for English and IndicBERT embeddings (128-dimensional contextual vectors) for Hindi. For words not found in the pre-trained vocabularies, random initialization is used. Both embeddings are fine-tuned during training to adapt better to the translation domain.

2.1 Model Architecture

Our model is based on the Seq2Seq framework with LSTM-based encoder and decoder components.

- **Encoder:** The encoder is a unidirectional LSTM that takes embedded English word sequences and produces hidden and cell states. GloVe embeddings are used to initialize the embedding layer.
- **Decoder:** The decoder is a unidirectional LSTM that generates Hindi words one at a time using the encoder's final hidden and cell states as the initial context. It uses IndicBERT embeddings as inputs. A linear layer maps LSTM outputs to the vocabulary space.
- **Teacher Forcing:** During training, the model uses teacher forcing with a configurable ratio to help stabilize learning by providing ground truth tokens as inputs to the decoder at certain steps.

2.2 Training Details

The model is implemented in PyTorch and trained on GPU. We use the following settings:

- Loss Function: CrossEntropyLoss (ignoring padding index)
- Optimizer: Adam
- Epochs: 100
- Batch Size: 64
- Embedding Dimensions: 100 (English), 128 (Hindi)
- Hidden Size: 1024
- Dropout: 0.5

Training is conducted using torchtext's `BucketIterator` to efficiently batch sentences of similar lengths, reducing padding and improving training speed. The model is validated periodically to monitor overfitting.

2.3 Evaluation and Results

2.3.1 Early Stopping

To prevent overfitting and ensure generalization, we used **early stopping** during training. We monitored the validation loss after every epoch, and training was halted if the validation loss did not improve for 5 consecutive epochs. This helped in avoiding unnecessary computation and reduced the risk of overfitting to the training data.

2.3.2 BLEU Scores

We evaluated our translation model using BLEU (Bilingual Evaluation Understudy) scores. These scores measure the overlap between the predicted translation and the reference translation using n-gram precision with a brevity penalty.

BLEU Scores on Training Set:

- BLEU@1: 17.72
- BLEU@2: 7.89
- BLEU@3: 3.67
- BLEU@4: 1.77

BLEU Scores on Test Set:

- BLEU@1: 18.09
- BLEU@2: 8.06
- BLEU@3: 3.79
- BLEU@4: 1.90

These values indicate that the model captures unigrams and bigrams reasonably well, while longer phrase matches are moderately preserved.

2.3.3 Loss Curve

The training and validation loss over the epochs are plotted in Figure 3.3. As seen from the figure, validation loss initially decreased along with training loss, but started to plateau, prompting early stopping.

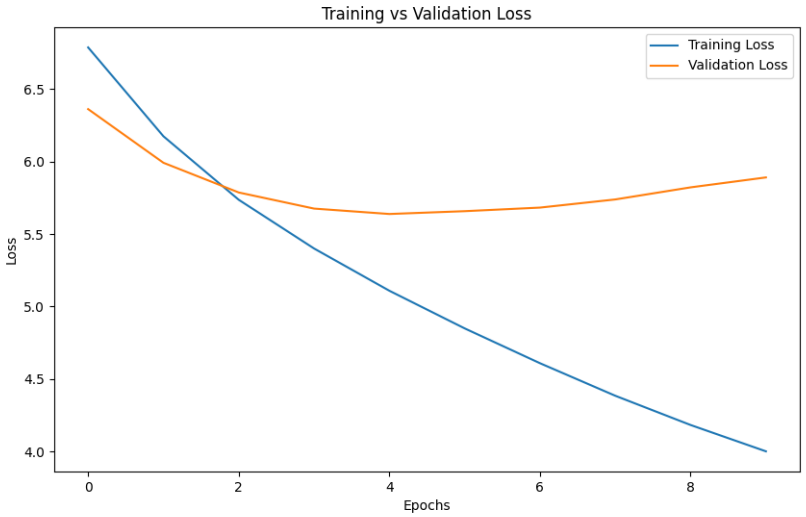


Figure 2.1: Training vs Validation Loss

2.3.4 Sample Translations

Sample translations for 5 train examples

```
Translations from Train Data:

Example 1:
  Source:      these cells usually repair themselves after chemotherapy .
  Target:      ये कोशिकाएं आमतौर पर रसोचिकित्सा के बाद अपनी मरम्मत कर लेती हैं।
  Translation: <unk> <unk> के <unk> में <unk> <unk>

Example 2:
  Source:      lord roche
  Target:      भगवान रोश
  Translation: <unk> रोश

Example 3:
  Source:      that time has gone .
  Target:      वह समय निकल गया है।
  Translation: यह भी भी हो

Example 4:
  Source:      sidbi will join to contribute to the transformation mission unleashed for these districts .
  Target:      सिडबी इन जिलों के लिए परावर्तन अभियान में योगदान देगा।
  Translation: <unk> के में के के के लिए के लिए के लिए के लिए

Example 5:
  Source:      i even called him today , told him that he should do a phd , she says .
  Target:      मैंने आज भी उन्हें फोन किया था कि उन्हें पीएचडी करनी चाहिए
  Translation: उन्होंने कहा कि कि यह यह <unk> था कि वह <unk> <unk>
```

Sample translations for 5 test examples

```
Translations from Test Data:

Example 1:
  Source:      it was the darkest day in the history of independent india .
  Target:      स्वतंत्र भारत के इतिहास में यह सबसे काला दिन था.
  Translation: यह भारत के भारत में भारत में है।

Example 2:
  Source:      those who die and leave wives behind should bequeath to them a year 's maintenance without
               causing them to leave their homes . but if they leave , no blame shall be attached to you in
               what they do with themselves kindly . allah is mighty and wise .
  Target:      उसी तरह ख़ुदा को याद करो और तुम में से जो लोग अपनी बीवियों को छोड़ कर मर जाएँ उन पर अपनी बीबियों के हक में साल भर तक के नान
               व नुफ़के (रोटी कपड़ा) और (घर से) न निकलने की वसीयत करनी (लाज़िम) है पस अगर औरतें ख़ुद निकल खड़ी हो तो जायज़ बातों (निकाह वगैरह)
               से कुछ अपने हक में करे उसका तुम पर कुछ इल्ज़ाम नही है और ख़ुदा हर यै पर ग़ालिब और हिकमत वाला है
  Translation: और वे लोग के <unk> <unk> और <unk> और <unk> और और <unk> और और <unk> और <unk>
               और और <unk> और <unk> और <unk> और <unk> और <unk> और <unk> और <unk> और और <unk> और <unk>
               और <unk> और <unk> और <unk> और <unk> <unk>

Example 3:
  Source:      sbi card is a subsidiary of state bank of india , the countrys largest lender .
  Target:      एसबीआई कैप भारतीय स्टेट बैंक की सहायक कंपनी है।
  Translation: <unk> बैंक बैंक बैंक बैंक बैंक बैंक के <unk> के एक <unk> है।

Example 4:
  Source:      spread awareness amongst the general public of india about green good deeds ,
               moefcc has prepared high quality audio - visual creatives ( 2 video spots : ggd plea and ggd
               montage 3 audio jingles : no plastic , air pollution and save water ) in collaboration with shri amitabh bachhan
  Target:      मवेशी जैव प्रौद्योगिकी गौजातीय पशुओं के लिए बोवाइन क्षय रोग (बी टीबी) के बारे में एक कार्यक्रम बिल एंड
               मेलिंडा गेट्स फाउंडेशन के सहयोग से शुरू किया गया, जिसमें बी टीबी प्रसार पर निगरानी रखने, बीसीजी टीकाकरण
               के जरिए बी टीबी नियंत्रण कार्यक्रम, रिपोजिटरी की स्थापना करने, और युवा वैज्ञानिकों के प्रशिक्षण पर ध्यान केंद्रित किया जाता है।
  Translation: भारत के के <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk>
               <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk>
               <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <unk> <
               unk> <unk> <unk> <unk>

Example 5:
  Source:      it brings peace and comfort to the family .
  Target:      इससे परिवार में सुख-शांति बनी रहती है।
  Translation: इससे परिवार और और और और और
```

2.4 Bidirectional LSTM

In Encoder, instead of unidirectional LSTM, we used bidirectional LSTM with hidden layer dimension as 512 and we concatenated outputs from 2 directions and passed it to unidirectional Decoder with hidden layer dimension as 1024 and other parameters are same.

BLEU Scores on Training Set:

- BLEU@1: 22.42
- BLEU@2: 11.29
- BLEU@3: 5.96
- BLEU@4: 3.30

BLEU Scores on Test Set:

- BLEU@1: 21.30
- BLEU@2: 10.28
- BLEU@3: 5.25
- BLEU@4: 2.90

There is slight and consistent improvement in the bleu scores of both train and test datasets.

3 Task 3: Machine translation using Transformer model

3.1 Introduction

The training dataset had 70000 instances comprising `source` (English sentence) and `target` (translated Hindi sentence), while validation dataset had 20000 instances in the same format.

We use pre-trained GloVe embeddings (200-dimensional vectors) for English and IndicBERT embeddings (768-dimensional contextual vectors) for Hindi. For words not found in the pre-trained vocabularies, random initialization is used. Both embeddings are fine-tuned during training to adapt better to the translation domain.

3.2 Model Architecture

Our model is based on the Seq2Seq framework with Transformer-based encoder and decoder components.

- **Encoder:** The encoder is designed to process input sequences and generate rich contextual representations. It consists of three identical layers that each perform the following operations:
  - **Self-Attention**: Enables the model to weigh the importance of different tokens in the input sequence relative to each other.
  - **Feedforward Network**: Applies a two-layer fully connected network to each token independently, enhancing feature representation.
  - **Layer Normalisation Dropout**: Improve training stability and generalisation by normalising activations and preventing overfitting.

  The encoder outputs a sequence of context-aware embeddings that are passed to the decoder.. GloVe embeddings are used to initialize the embedding layer.

```
TransformerEncoder(
  (layers): ModuleList(
    (0-2): 3 x TransformerEncoderLayer(
      (self_attn): MultiheadAttention(
        (out_proj): NonDynamicallyQuantizableLinear(in_features=256, out_features=256, bias=True)
      )
      (linear1): Linear(in_features=256, out_features=512, bias=True)
      (dropout): Dropout(p=0.1, inplace=False)
      (linear2): Linear(in_features=512, out_features=256, bias=True)
      (norm1): LayerNorm((256,), eps=1e-05, elementwise_affine=True)
      (norm2): LayerNorm((256,), eps=1e-05, elementwise_affine=True)
      (dropout1): Dropout(p=0.1, inplace=False)
      (dropout2): Dropout(p=0.1, inplace=False)
    )
  )
  (norm): LayerNorm((256,), eps=1e-05, elementwise_affine=True)
)
```

Figure 3.1: Encoder module architecture

- **Decoder:** The decoder generates output sequences (e.g., translations or predictions) based on encoder outputs and previously generated tokens. It also has three layers, each with the following components:
  - **Masked Self-Attention**: Allows the decoder to attend only to earlier positions in the output sequence, preserving the autoregressive property.
  - **Cross-Attention**: Lets the decoder attend to the encoder's output, aligning input and output sequences.
  - **Feedforward Network**: Transforms features at each position to refine the output representations.
  - **Layer Normalisation Dropout**: Enhance training robustness and reduce overfitting.

The decoder produces the final output sequence one step at a time, using attention mechanisms to integrate both past outputs and encoded input information. It uses IndicBERT embeddings as inputs. A linear layer maps outputs to the vocabulary space.

```
TransformerDecoder(
  (layers): ModuleList(
    (0-2): 3 x TransformerDecoderLayer(
      (self_attn): MultiheadAttention(
        (out_proj): NonDynamicallyQuantizableLinear(in_features=256, out_features=256, bias=True)
      )
      (multihead_attn): MultiheadAttention(
        (out_proj): NonDynamicallyQuantizableLinear(in_features=256, out_features=256, bias=True)
      )
      (linear1): Linear(in_features=256, out_features=512, bias=True)
      (dropout): Dropout(p=0.1, inplace=False)
      (linear2): Linear(in_features=512, out_features=256, bias=True)
      (norm1): LayerNorm((256,), eps=1e-05, elementwise_affine=True)
      (norm2): LayerNorm((256,), eps=1e-05, elementwise_affine=True)
      (norm3): LayerNorm((256,), eps=1e-05, elementwise_affine=True)
      (dropout1): Dropout(p=0.1, inplace=False)
      (dropout2): Dropout(p=0.1, inplace=False)
      (dropout3): Dropout(p=0.1, inplace=False)
    )
  )
  (norm): LayerNorm((256,), eps=1e-05, elementwise_affine=True)
)
```

Figure 3.2: Decoder module architecture

3.3 Training Details

The model is implemented in PyTorch and trained on GPU. We use the following settings:

- Loss Function: Cross Entropy Loss
- Optimizer: Adam
- Epochs: 20
- Batch Size: 16
- Embedding Dimensions: 200 (English), 768 (Hindi)
- Number of heads: 8
- Number of layers: 3
- Feedforward Dimensions: 512
- Dropout: 0.1
- Learning rate: 1e-04

3.4 Results

3.4.1 BLEU Scores

We evaluated our translation model using BLEU (Bilingual Evaluation Understudy) scores. These scores measure the overlap between the predicted translation and the reference translation using n-gram precision with a brevity penalty.

- BLEU-1 (Unigram): 0.1330

- BLEU-2 (Bigram): 0.0696
- BLEU-3 (Trigram): 0.0386
- BLEU-4 (4-gram): 0.0223

3.4.2 Loss Curve

The training and validation loss over the epochs are plotted below

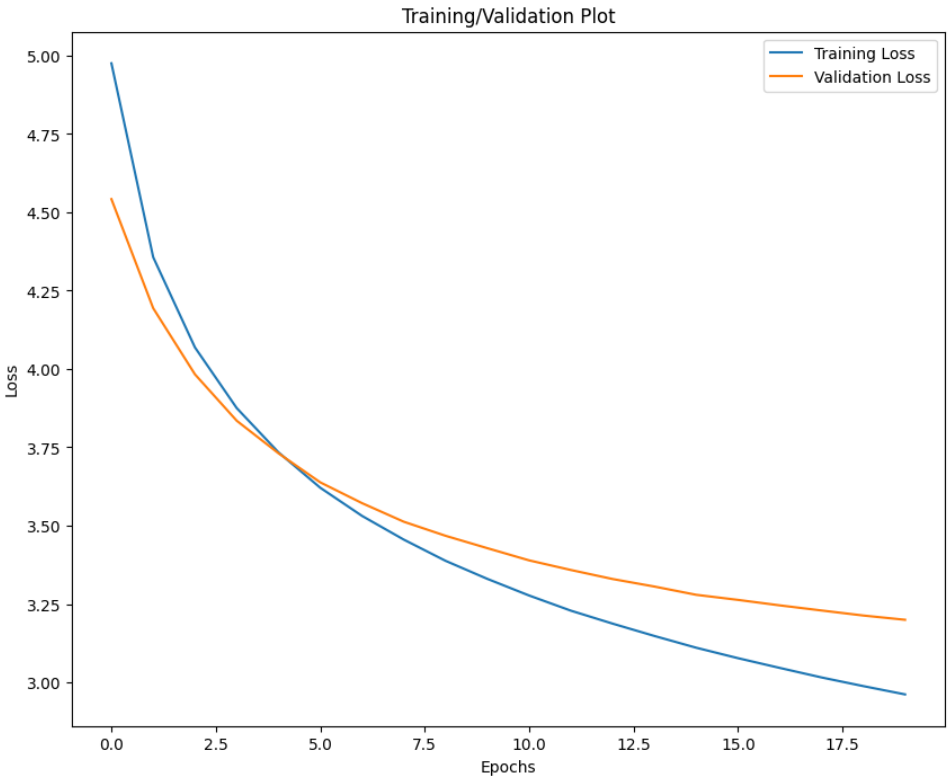
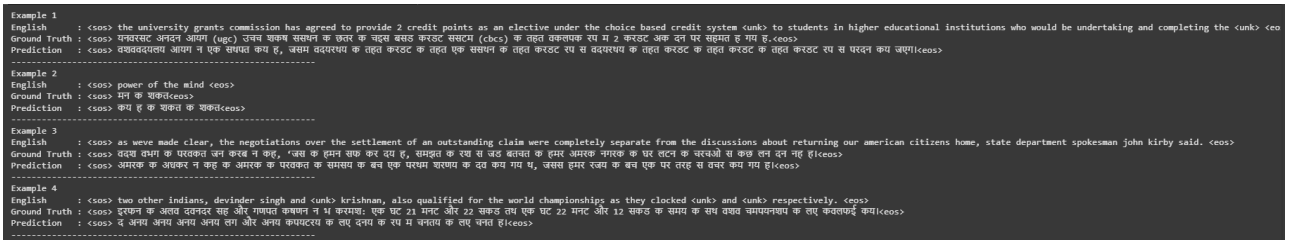


Figure 3.3: Training vs Validation Loss

3.4.3 Sample Translations



3.5 Conclusions

The Transformer-based model was trained for 20 epochs with a learning rate of **1e-4** and a **dropout rate of 0.1**, using limited GPU resources. The training and validation loss curves show a consistent downward trend with no signs of overfitting, indicating that the model was learning effectively within its capacity and computational constraints.

However, BLEU scores on the test set remain relatively low, particularly for higher n-grams These scores suggest that while the model is learning to generate somewhat relevant words, it struggles with producing coherent or fluent multi-word sequences, which is critical for high-quality natural

language generation.
Nonetheless, the training could have gone beyond 20 epochs until we hit callback for more optimised model!

3.6 Scope of improvement

- **Increase Model Capacity**:
  - Add more encoder/decoder layers
  - Increase the model's hidden size and number of attention heads
- **Beam Search Decoding**: At inference time, replace greedy decoding with beam search to improve fluency and BLEU scores. heads