# Transformers

Zeham Management Technologies BootCamp
by SDAIA

September 12th, 2024

SDAIA
الهيئة السعودية للبيانات
والذكاء الاصطناعي
Saudi Data & AI Authority

# Agenda

- Fine-tuning
- Advanced Technologies

# Fine-Tuning

# Fine-Tuning Transformers

Use pre-trained models as a starting point for custom tasks, reducing training time and resources.

Fine-Tuning Workflow:

1. Load pretrained model.
2. Prepare dataset (tokenization, formatting).
3. Use Hugging Face's Trainer API for training.

# Hyperparameter Tuning in Fine-Tuning

Fine-tuning requires careful selection of hyperparameters such as learning rate, batch size, and the number of training epochs.

- **Learning Rate:** Adjusting learning rates can prevent overfitting or underfitting.
- **Batch Size:** Smaller batch sizes give more granular updates but take longer, whereas larger batches improve training speed but may generalize less well.
- **Number of Epochs:** This determines how many times the model will pass through the entire dataset during training.

# Learning Rate Scheduling

Transformers have many layers and parameters, making them sensitive to learning rate. Adjusting it dynamically during fine-tuning can lead to better convergence and stability.

## Popular Learning Rate Schedulers for Transformers:

- **Warmup with Linear Decay:** Often used in transformers, this technique increases the learning rate gradually at the start (warmup) and then decreases it linearly. Effective for large pre-trained models.
- **Cosine Annealing:** Reduces the learning rate in a cyclic manner, useful for longer fine-tuning tasks where the model requires smooth convergence.

# Using Dropout for Transformer Regularization

## What is Dropout in Transformers?

Dropout is a regularization technique where some neurons are "dropped out" or turned off during each forward pass to prevent overfitting.

In transformers, dropout can be applied to attention layers and feed-forward layers to ensure that the model generalizes well to unseen data during fine-tuning.

## When to Use Dropout in Fine-Tuning:

- Dropout is especially useful when fine-tuning on smaller datasets where overfitting is a concern.
- Applying it to the attention weights ensures that the model doesn't rely too heavily on specific attention patterns.

# Layer Freezing for Efficient Fine-Tuning

## Why Freeze Layers in Transformers?

Freezing the lower layers of a transformer model can drastically speed up fine-tuning, especially when those layers capture general features from the pretraining process.

By freezing lower layers and only updating the top layers, you reduce computation while focusing learning on task-specific patterns.

## When to Use Layer Freezing:

Effective when transferring a transformer model to a new task with limited data, where task-specific features are more important than general language understanding.

Often, only the last few layers are fine-tuned to adapt to the new task, especially for BERT-like models.

# Layer Freezing for Efficient Fine-Tuning

## Freeze Layers in Transformers Example:

**Code Example (Freezing Layers in Hugging Face Transformers):**

python                                                                                    Copy code

```python
# Freeze all layers except the last
for param in model.bert.encoder.layer[:10].parameters():
    param.requires_grad = False
```

# Handling Imbalanced Datasets in Fine-Tuning

## Why Imbalanced Data Matters:

Transformers can overfit to majority classes when fine-tuning on imbalanced datasets.

Addressing this improves model generalization to minority classes.

## Strategies:

- **Oversampling Minority Class:** Duplicate samples from the minority class to balance the dataset.
- **Class Weights:** Assign higher loss weights to the minority class to balance the contribution to the loss function.

# Handling Imbalanced Datasets in Fine-Tuning

## Class Weights Example:

**Code Example (Using Class Weights in Transformers):**

python

Copy code

```python
from torch.nn import CrossEntropyLoss

# Define weights for imbalanced classes
class_weights = torch.tensor([0.7, 1.3])

# Use class weights in the loss function
loss_fct = CrossEntropyLoss(weight=class_weights)
```

# Advanced Technologies

# Advanced Optimization Algorithms

- **AdamW:** Combines adaptive learning rates with weight decay, commonly used in fine-tuning.
- **RMSprop:** Effective when training on highly noisy or non-stationary data.
- **SGD with Momentum:** A variant of stochastic gradient descent that accelerates training by keeping a "momentum" term.

# Advanced Optimization Algorithms

## AdamW Example:

python

Copy code

```python
from transformers import AdamW

# Define the optimizer using AdamW
optimizer = AdamW(model.parameters(), lr=5e-5, weight_decay=0.01)

# Training loop
for batch in train_dataloader:
    outputs = model(**batch)
    loss = outputs.loss
    loss.backward()

    optimizer.step()
    optimizer.zero_grad()
```

# Mixed Precision Training for Faster Fine-Tuning

## What is Mixed Precision Training?

Mixed precision involves using both 16-bit and 32-bit floating point types during training to speed up computation and reduce memory usage without sacrificing much accuracy.

## Advantages for Transformers:

- Faster training time with less memory usage, especially useful for large models like BERT, GPT-2, or T5.
- Hugely beneficial when training on limited hardware resources.

# Mixed Precision Training for Faster Fine-Tuning

## Mixed Precision Example:

**Code Example (Using Hugging Face's Trainer API with Mixed Precision):**

python Copy code

```python
from transformers import Trainer, TrainingArguments

training_args = TrainingArguments(
    fp16=True,  # Enable mixed precision training
    per_device_train_batch_size=16,
    ...
)

trainer = Trainer(
    model=model,
    args=training_args,
    ...
)
```

# Tutorial

8-Transformers/LAB/ Question Answering with Transformers.ipynb

# Tutorial

8-Transformers/LAB/ Text-to-Text Generation.ipynb

# Tutorial

8-Transformers/LAB/ Fill Mask.ipynb

# Thank you!

SDAIA
الهيئة السعودية للبيانات والذكاء الاصطناعي
Saudi Data & AI Authority