Fine-tuning BERT for Text Classification

Overview with Example Code

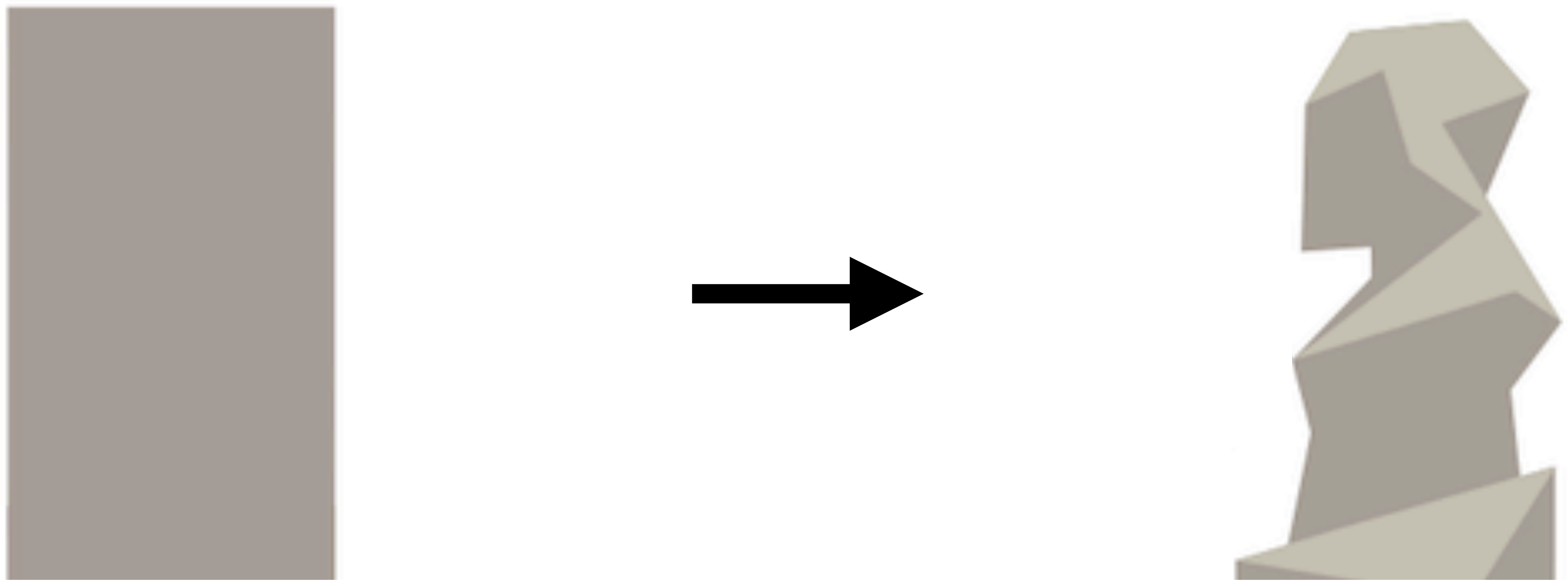
Shaw Talebi

Fine-tuning BERT for Text Classification

1 2 3

Fine-tuning

Adapting a pre-trained model to a particular task (through additional training)



Pre-trained Model

(Self-supervised)

~3B Words

(BERT)

Fine-tuned Model

(Supervised)

~3k Examples

(MRCP)

[1]

Fine-tuning

Adapting a pre-trained model to a particular task (through additional training)

Fine-tuned Model

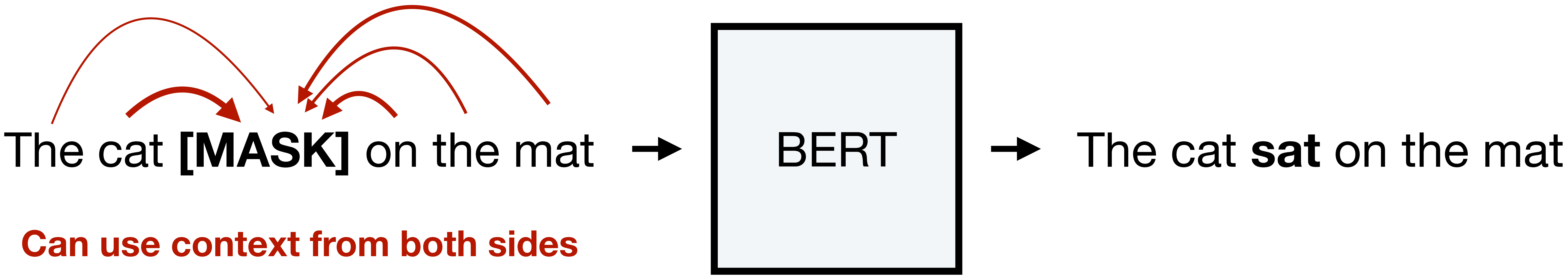
Additional Fine-tuning

[1]

BERT

Language model developed for fine-tuning

Task #1: Masked LM



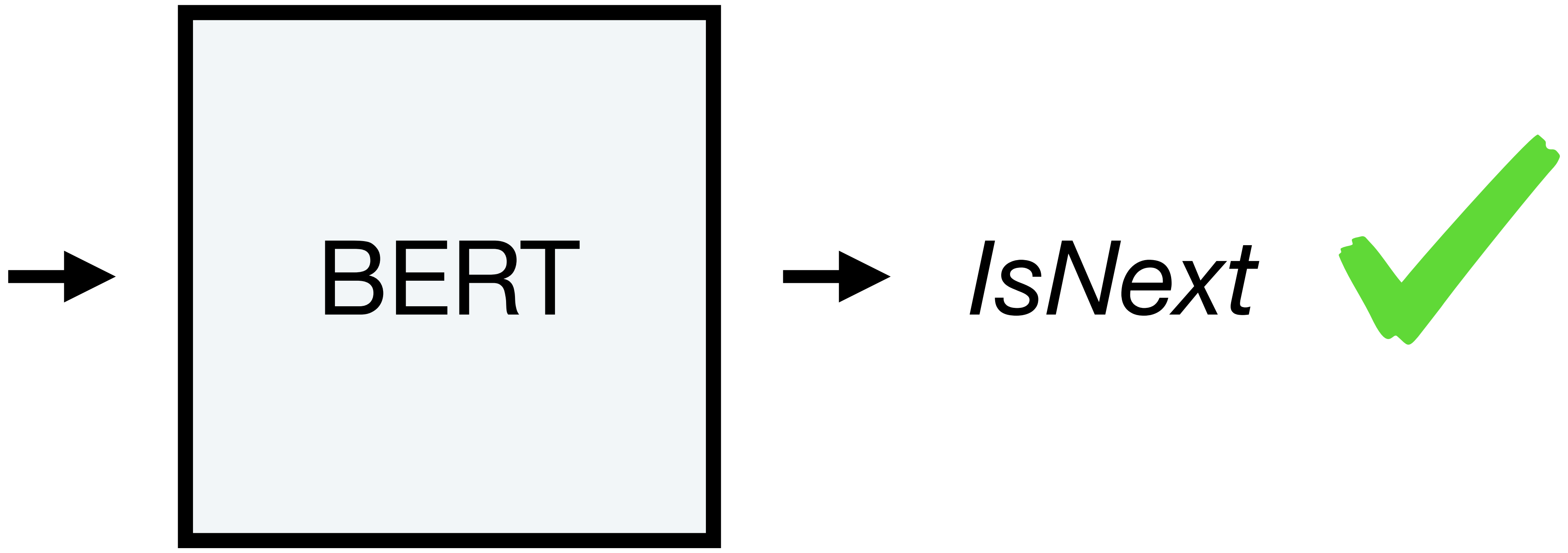
[1]

BERT

Language model developed for fine-tuning

Task #2: Next Sentence Prediction

[CLS] BERT is conceptually simple and empirically powerful. **[SEP]** It obtains new state-of-the-art results on 11 NLP tasks **[SEP]**



[1]

BERT

Language model developed for fine-tuning

Task #2: Next Sentence Prediction

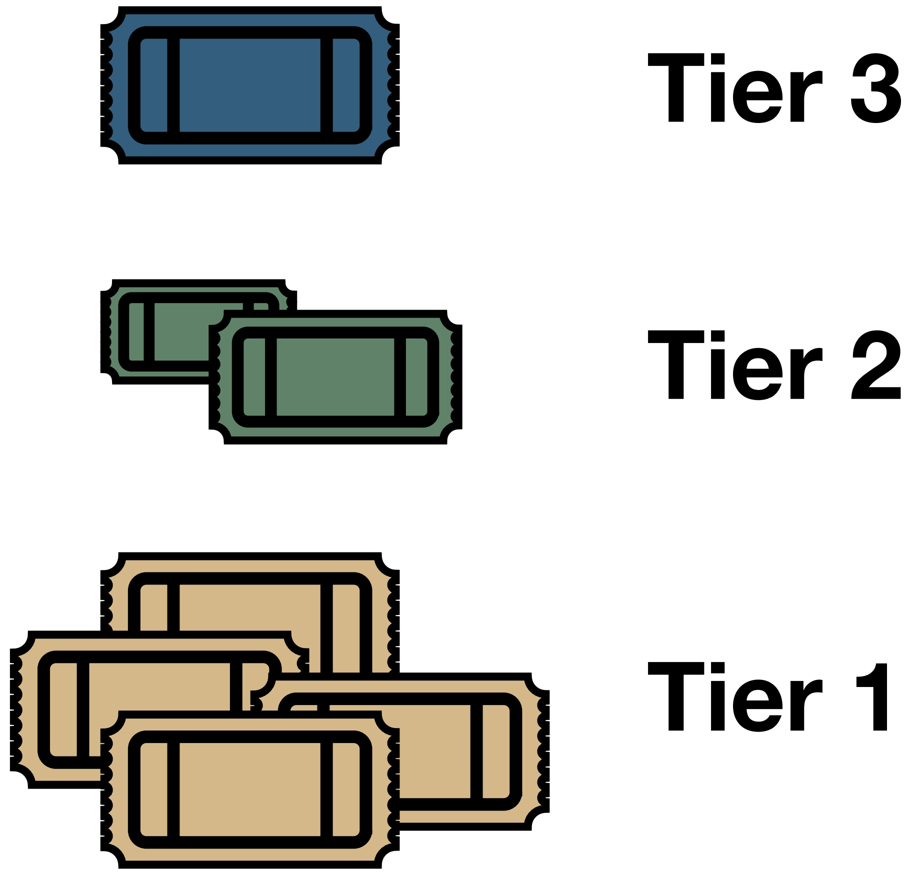
[CLS] BERT is conceptually simple and empirically powerful. **[SEP]** The cat sat on the mat **[SEP]** → BERT → *NotNext* ✗

[1]

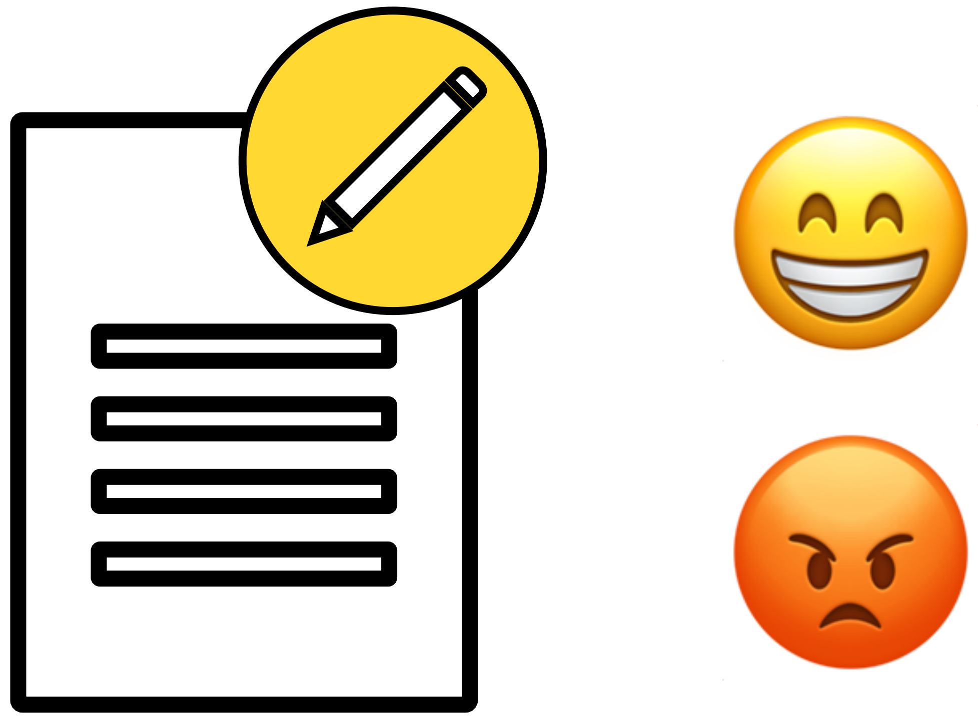
Text Classification

Assigning a label to text sequences

Spam Detection



Categorizing IT Tickets

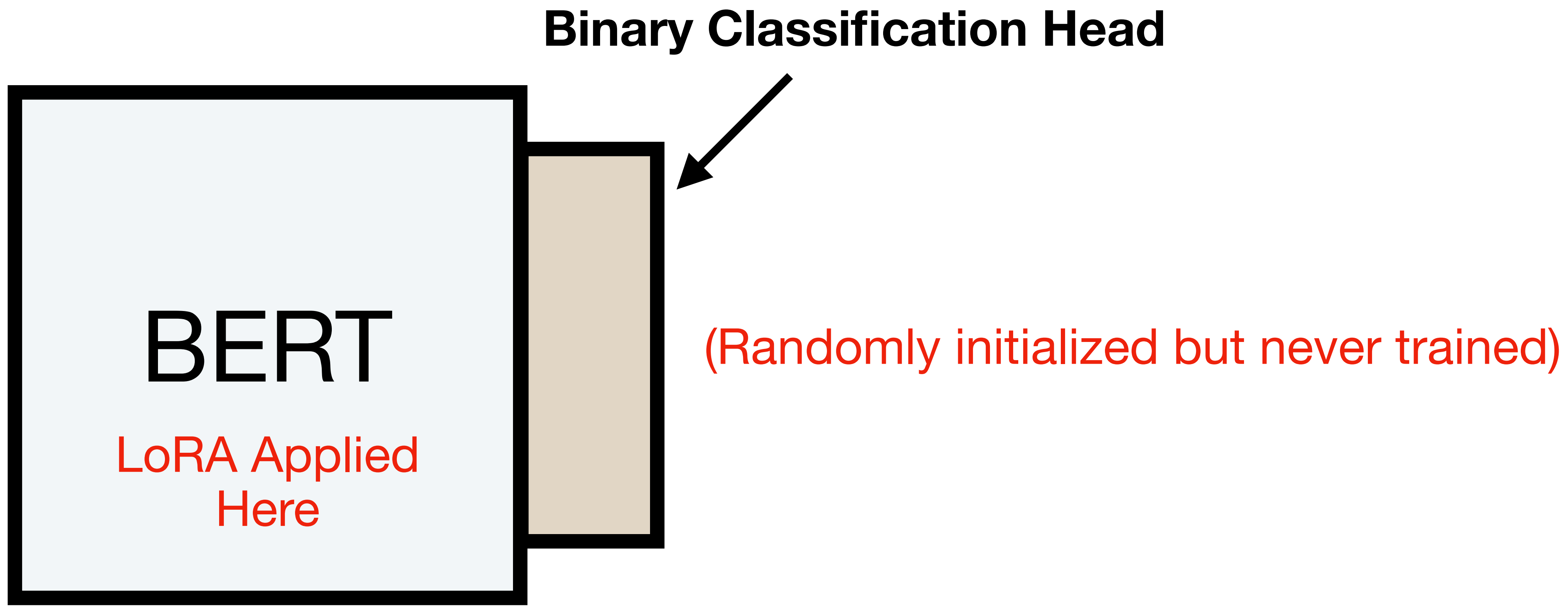


Review Sentiment
Analysis

Example Code: Fine-tuning BERT for Phishing URL Identification

Epoch	Training Loss	Validation Loss	Accuracy
1	No log	0.423429	{'accuracy': 0.876}
2	0.412400	0.551401	{'accuracy': 0.878}
3	0.412400	0.593060	{'accuracy': 0.899}
4	0.211600	0.640572	{'accuracy': 0.894}
5	0.211600	0.839775	{'accuracy': 0.891}
6	0.064300	0.930830	{'accuracy': 0.887}
7	0.064300	0.988515	{'accuracy': 0.889}
8	0.020000	1.007572	{'accuracy': 0.884}
9	0.020000	1.040836	{'accuracy': 0.888}
10	0.009500	1.034907	{'accuracy': 0.896}

Overfitting



Example Code: Fine-tuning BERT for Phishing URL Identification

Imports

```python
from datasets import DatasetDict, Dataset
from transformers import AutoTokenizer, AutoModelForSequenceClassification,
                         TrainingArguments, Trainer
import evaluate
import numpy as np
from transformers import DataCollatorWithPadding
```

Load Data

```python
dataset_dict = load_dataset("shawhin/phishing-site-classification")
```

[https://huggingface.co/datasets/shawhin/phishing-site-classification]

https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs/model-compression

Example Code: Fine-tuning BERT for Phishing URL Identification

Load Pre-trained Model

```
# define pre-trained model path
model_path = "google-bert/bert-base-uncased"

# load model tokenizer
tokenizer = AutoTokenizer.from_pretrained(model_path)

# load model with binary classification head
id2label = {0: "Safe", 1: "Not Safe"}
label2id = {"Safe": 0, "Not Safe": 1}
model = AutoModelForSequenceClassification.from_pretrained(model_path,
                                                           num_labels=2,
                                                           id2label=id2label,
                                                           label2id=label2id,)
```


https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs/model-compression

Example Code: Fine-tuning BERT for Phishing URL Identification

Set Trainable Parameters

```
# freeze all base model parameters
for name, param in model.base_model.named_parameters():
    param.requires_grad = False

# unfreeze base model pooling layers
for name, param in model.base_model.named_parameters():
    if "pooler" in name:
        param.requires_grad = True
```


https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs/model-compression

Example Code: Fine-tuning BERT for Phishing URL Identification

Data Pre-processing

```
# define text preprocessing
def preprocess_function(examples):
    # return tokenized text with truncation
    return tokenizer(examples["text"], truncation=True)

# preprocess all datasets
tokenized_data = dataset_dict.map(preprocess_function, batched=True)
```

```
# create data collator
data_collator = DataCollatorWithPadding(tokenizer=tokenizer)
```


https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs/model-compression

Example Code: Fine-tuning BERT for Phishing URL Identification

Define Evaluation Metrics

```python
# load metrics
accuracy = evaluate.load("accuracy")
auc_score = evaluate.load("roc_auc")

def compute_metrics(eval_pred):
    # get predictions
    predictions, labels = eval_pred

    # apply softmax to get probabilities
    probabilities = np.exp(predictions) / np.exp(predictions).sum(-1,
                                                              keepdims=True)
    # use probabilities of the positive class for ROC AUC
    positive_class_probs = probabilities[:, 1]
    # compute auc
    auc = np.round(auc_score.compute(prediction_scores=positive_class_probs,
                                     references=labels)['roc_auc'],3)

    # predict most probable class
    predicted_classes = np.argmax(predictions, axis=1)
    # compute accuracy
    acc = np.round(accuracy.compute(predictions=predicted_classes,
                                     references=labels)['accuracy'],3)

    return {"Accuracy": acc, "AUC": auc}
```


https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs/model-compression

Example Code: Fine-tuning BERT for Phishing URL Identification

Training Parameters

```
# hyperparameters
lr = 2e-4
batch_size = 8
num_epochs = 10

training_args = TrainingArguments(
    output_dir="bert-phishing-classifier_teacher",
    learning_rate=lr,
    per_device_train_batch_size=batch_size,
    per_device_eval_batch_size=batch_size,
    num_train_epochs=num_epochs,
    logging_strategy="epoch",
    eval_strategy="epoch",
    save_strategy="epoch",
    load_best_model_at_end=True,
)
```


https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs/model-compression

Example Code: Fine-tuning BERT for Phishing URL Identification

Fine-tune Model

```
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_data["train"],
    eval_dataset=tokenized_data["test"],
    tokenizer=tokenizer,
    data_collator=data_collator,
    compute_metrics=compute_metrics,
)

trainer.train()
```

Training Loss	Epoch	Step	Validation Loss	Accuracy	Auc
0.4916	1.0	263	0.4228	0.784	0.915
0.3894	2.0	526	0.3586	0.818	0.932
0.3837	3.0	789	0.3144	0.86	0.939
0.3574	4.0	1052	0.4494	0.807	0.942
0.3517	5.0	1315	0.3287	0.86	0.947
0.3518	6.0	1578	0.3042	0.871	0.949
0.3185	7.0	1841	0.2900	0.862	0.949
0.3267	8.0	2104	0.2958	0.876	0.95
0.3153	9.0	2367	0.2881	0.867	0.951
0.3061	10.0	2630	0.2963	0.873	0.951

https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs/model-compression

Example Code: Fine-tuning BERT for Phishing URL Identification

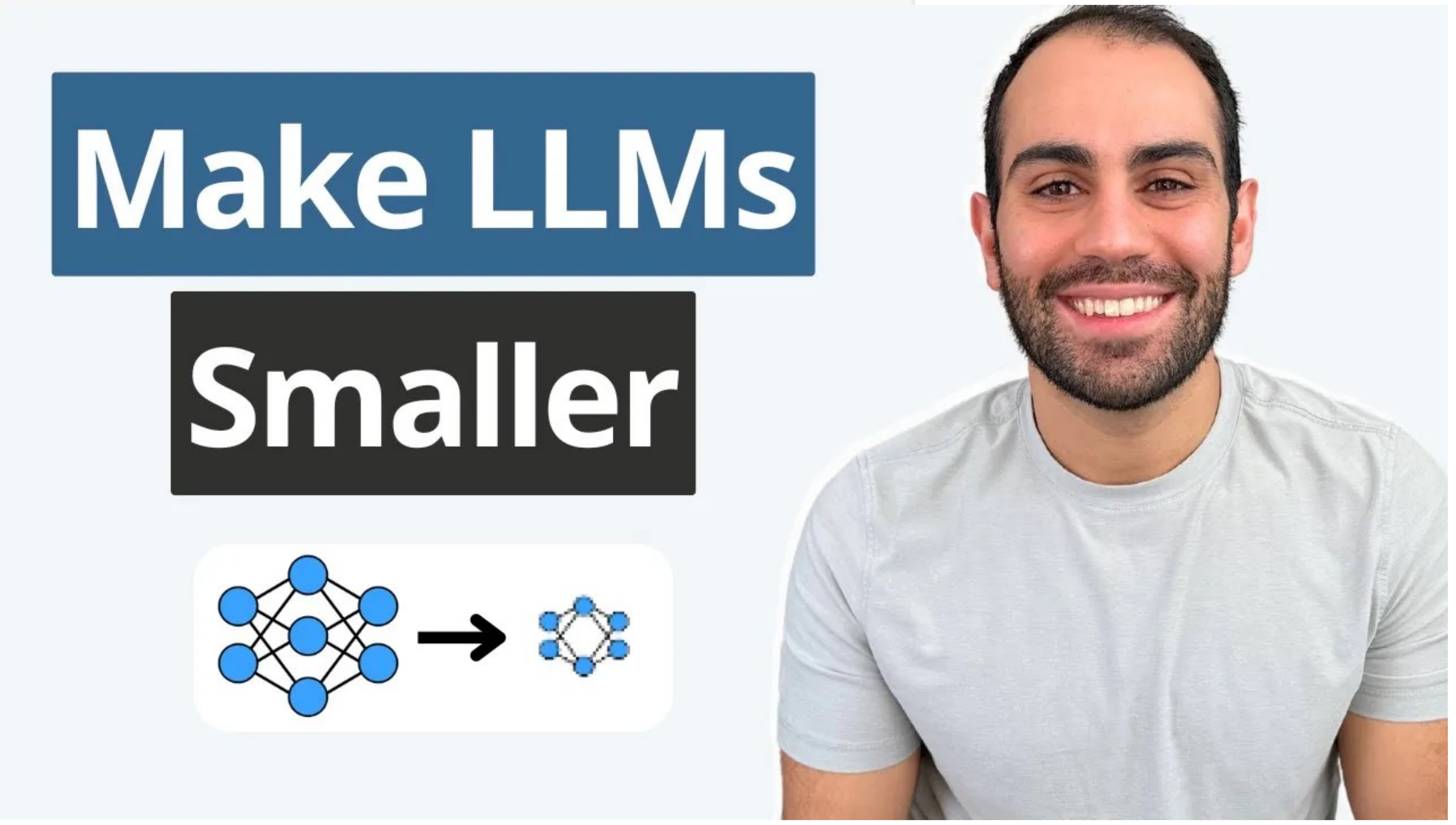
Validation Data

```
# apply model to validation dataset
predictions = trainer.predict(tokenized_data["validation"])

# Extract the logits and labels from the predictions object
logits = predictions.predictions
labels = predictions.label_ids

# Use your compute_metrics function
metrics = compute_metrics((logits, labels))
print(metrics)

# >> {'Accuracy': 0.889, 'AUC': 0.946}
```



https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs/model-compression

Fine-tuning BERT for Text Classification

A Hackable Example with Python Code

Although today's 100B+ parameter transformer models are state-of-the-art in AI, there's still much we can accomplish with smaller (< 1B parameter) models. In this article, I will walk through one such example, fine-tuning BERT (110M parameters) to classify phishing URLs. I'll start by covering key concepts and then share example Python code.

Friend Link in Description

👇