Your grade: 100%

Your latest: **100%** • Your highest: **100%** • To pass you need at least 80%. We keep your highest score.

Next item

1. Which of the following reasons makes PyTorch an excellent platform for rapid prototyping and expedites debugging? 1 / 1 point

- It offers essential tools for creating a variety of machine learning models.
- (selected) It allows running and testing portions of code in real time rather than waiting for the entire code to be implemented.
- It supports a wide variety of neural network architectures and offers extensive libraries of pre-configured (and even pre-trained) models.
- It has an intuitive and straightforward syntax, which is Python-based.

Correct
PyTorch allows data scientists to run and test portions of code in real time rather than waiting for the entire code to be implemented, which can take a very long time for large deep-learning models. This makes PyTorch an excellent platform for rapid prototyping and greatly expedites debugging.

2. Match the pitfalls of fine-tuning with the correct description to avoid them and select the correct option. 1 / 1 point

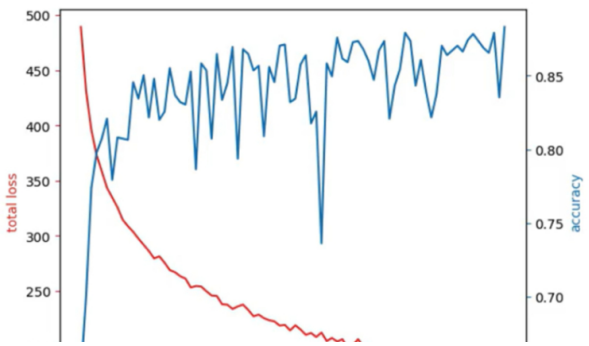
	Pitfall		How to avoid
1.	Overfitting	a.	Ensure sufficient training and an appropriate learning rate
2.	Underfitting	b.	Keep training and validation datasets separate
3.	Catastrophic forgetting	c.	Avoid using a small dataset or extending training epochs excessively
4.	Data leakage	d.	Prevent the model from losing its initial broad knowledge

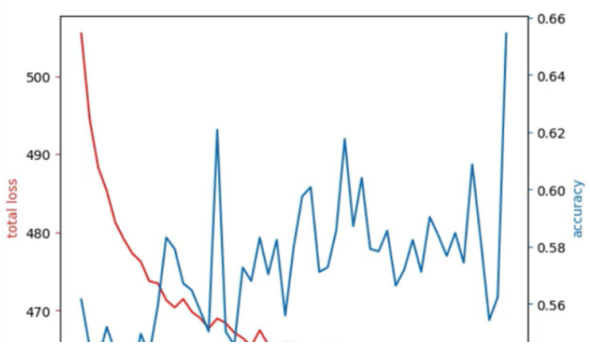
- (selected) 1-c; 2-a; 3-d; 4-b
- 1-d; 2-c; 3-a; 4-b
- 1-c; 2-b; 3-d; 4-a
- 1-a; 2-c; 3-d; 4-b

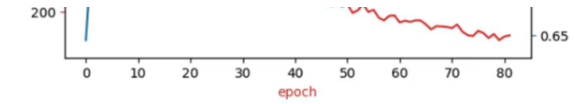
Correct
To avoid pitfalls in fine-tuning large language models (LLMs), you should consider the following:

- Overfitting: To prevent the model from performing well only on training data, avoid using a small dataset or extending training epochs excessively.
- Underfitting: Ensure sufficient training and an appropriate learning rate to enable adequate learning.
- Catastrophic forgetting: Prevent the model from losing its initial broad knowledge, which can hinder performance on various NLP tasks.
- Data Leakage: Keep training and validation datasets separate to avoid misleading metrics.

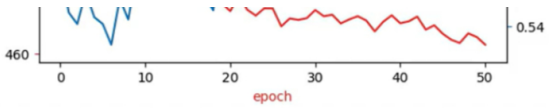
3. Consider the following plots A and B. Which of the given statements is correct regarding the plots? 1 / 1 point







Plot A



Plot B

- (●) Plot A shows the loss versus accuracy of the completely fine-tuned model, and plot B shows the loss versus accuracy of the model fine-tuned only on the final layer.
- ( ) Plot A shows the loss versus accuracy of the model fine-tuned only on the final layer, and plot B shows the loss versus accuracy of the completely fine-tuned model.
- ( ) Plot A shows the loss versus accuracy of the model fine-tuned only on a few parameters, and plot B shows the loss versus accuracy of the completely fine-tuned model.
- ( ) Plot A shows the loss versus accuracy of a pre-trained but not a fine-tuned model, and plot B shows the loss versus accuracy of the model fine-tuned only on a few parameters.

Correct
Plot A: This plot shows the loss versus accuracy of the model completely fine-tuned, which achieves approximately 90% accuracy on the validation data.

Plot B: This plot shows the loss versus accuracy of the model fine-tuned only on the final layer. While training is much faster, the performance is significantly worse.

4. Consider the following code: 1 / 1 point

```
tokenized_datasets['train'][0].keys():
dict_keys(['label', 'text', 'input_ids', 'token_type_ids', 'attention_mask'])

tokenized_datasets = tokenized_datasets.remove_columns(["text"])
tokenized_datasets = tokenized_datasets.rename_column("label", "labels")
tokenized_datasets.set_format("torch")
```

Which of the following options correctly explains the running of the code?

- ( ) The code removes the “text” key, renames the “label” key, and converts data into layers.
- ( ) The code only removes the “text” key and renames the “label” key.
- (●) The code removes the “text” key, renames the “label” key, and converts data into PyTorch tensors.
- ( ) The code renames the “text” key, removes the “label” key, and converts data into embeddings.

Correct
As the model does not need text information, you can remove the “text” and rename the label key. The data is then converted into PyTorch tensors.

The result is a set of tensors with the keys: “labels”, “input_ids”, “token_type_ids”, and “attention_mask”. The resulting code is shown below:

```
tokenized_datasets['train'][0].keys():
dict_keys(['labels', 'input_ids', 'token_type_ids', 'attention_mask'])
```

5. Consider the following code snippet: 1 / 1 point

```
from transformers import pipeline
```

```
text = "This is a [MASK] movie!"
mask_filler = pipeline(task='fill-mask', model=model,tokenizer=tokenizer)
```

```
results = mask_filler(text)
for result in results:
   print(f"Predicted token: {result['token_str']}, Confidence: {result['score']:.2f}")
```

Which of the following will be the output of the given code snippet?

◉
```
Predicted token: great, Confidence: 0.21
Predicted token: new, Confidence: 0.08
Predicted token: good, Confidence: 0.06
Predicted token: fantastic, Confidence: 0.05
Predicted token: wonderful, Confidence: 0.05
```

○
```
Predicted token: wonderful, Confidence: 0.05
Predicted token: fantastic, Confidence: 0.05
Predicted token: good, Confidence: 0.06
Predicted token: new, Confidence: 0.08
Predicted token: great, Confidence: 0.21
```

○
```
Predicted token: fantastic, Confidence: 0.05
Predicted token: wonderful, Confidence: 0.05
Predicted token: good, Confidence: 0.06
Predicted token: new, Confidence: 0.08
Predicted token: great, Confidence: 0.21
```

○
```
Predicted token: great, Confidence: 0.21
```

Correct

"mask filler" is a pipeline object created to make predictions and the parameter "task" specifies the problem type. The input includes the model and the tokenizer, which allows you to tokenize the data and make a prediction in one step.

The output result for the input text "This is a [MASK] movie" will be an iterable object. The key token_str will contain the predicted token value for the mask, and the key "score" will indicate the likelihood.

The samples will be ordered based on the order of likelihood. Observing the output, you can see that "great" is the most likely token.

```
Predicted token: great, Confidence: 0.21
Predicted token: new, Confidence: 0.08
Predicted token: good, Confidence: 0.06
Predicted token: fantastic, Confidence: 0.05
Predicted token: wonderful, Confidence: 0.05
```