Here's a detailed explanation of **Advanced Transformer Models** — **BERT**, **GPT**, and **T5** — including their core architectures, mathematical intuition, and how to **fine-tune** them for downstream tasks, with **code examples** in PyTorch using the Hugging Face transformers library.

✅ Class 17: Advanced Transformer Models

🎯 Learning Objectives Recap

- Understand the architecture and functioning of **BERT**, **GPT**, and **T5**
- Understand the **mathematical foundation** behind each model
- Learn how to **fine-tune** these models for classification, generation, and translation

---

🔷 1. Transformer Recap (Foundation)

Key Components:

- **Input Embeddings** + **Positional Encodings**
- **Multi-Head Self-Attention**:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

- **Feed-Forward Networks (FFN)**:

$$\text{FFN}(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2$$

🔶 2. BERT (Bidirectional Encoder Representations from Transformers)

🔸 Architecture:

- Only **Encoder** stack
- **Bidirectional** self-attention
- Pre-trained on:
    - **Masked Language Modeling (MLM)**

$$\text{Loss}_{MLM} = - \sum_{i \in M} \log P(x_i | x_{\setminus i})$$

    - **Next Sentence Prediction (NSP)**

🔸 Input Format:

[CLS] Sentence A [SEP] Sentence B [SEP]

🔸 **Fine-Tuning: Text Classification Example**

```
from transformers import BertTokenizer, BertForSequenceClassification, Trainer, TrainingArguments
from datasets import load_dataset

# Load dataset and tokenizer
dataset = load_dataset("imdb")
tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")

def tokenize_function(examples):
    return tokenizer(examples['text'], padding="max_length", truncation=True)

encoded_dataset = dataset.map(tokenize_function, batched=True)
model = BertForSequenceClassification.from_pretrained("bert-base-uncased", num_labels=2)

# Fine-tuning
training_args = TrainingArguments(output_dir="./results", evaluation_strategy="epoch", per_device_train_batch_size=8)
trainer = Trainer(model=model, args=training_args, train_dataset=encoded_dataset['train'], eval_dataset=encoded_dataset['test'])
trainer.train()
```

🔷 **3. GPT (Generative Pretrained Transformer)**

🔸 **Architecture:**

- Only **Decoder** stack
- **Unidirectional (causal)** attention:

Attention mask: Prevent future token access\text{Attention mask: Prevent future token access}

🔸 **Pre-training:**

- **Causal Language Modeling (CLM)**:

$$\text{Loss}_{CLM} = - \sum_t \log P(x_t | x_{<t})$$

🔸 **Fine-Tuning: Text Generation**

```
from transformers import GPT2Tokenizer, GPT2LMHeadModel

tokenizer = GPT2Tokenizer.from_pretrained("gpt2")
model = GPT2LMHeadModel.from_pretrained("gpt2")

input_ids = tokenizer.encode("Once upon a time", return_tensors="pt")
output = model.generate(input_ids, max_length=50, do_sample=True)

print(tokenizer.decode(output[0], skip_special_tokens=True))
```

🔷 **4. T5 (Text-To-Text Transfer Transformer)**

🔸 **Architecture:**

- **Encoder-Decoder** (like original Transformer)
- Every task is cast as **text-to-text**

🔸 **Pre-training:**

- **Span Corruption (Masked Span Prediction)**:

$$\text{Input:} \quad \text{“The capital of <extra_id_0> is <extra_id_1>.”} \quad \text{Target:} \quad \text{“<extra_id_0> France <extra_id_1> Paris”}$$

🔸 **Fine-Tuning: Translation Example**

```
from transformers import T5Tokenizer, T5ForConditionalGeneration

tokenizer = T5Tokenizer.from_pretrained("t5-small")
model = T5ForConditionalGeneration.from_pretrained("t5-small")

input_text = "translate English to French: The house is wonderful."
input_ids = tokenizer(input_text, return_tensors="pt").input_ids
```

output_ids = model.generate(input_ids)

print(tokenizer.decode(output_ids[0], skip_special_tokens=True))

🧠 Mathematical View Comparison

Model	Direction	Objective	Equation
BERT	Bidirectional	Masked LM + NSP	Loss=LossMLM+LossNSP\text{Loss} = \text{Loss}_{MLM} + \text{Loss}_{NSP}
GPT	Unidirectional	Causal LM	(\text{Loss} = -\sum_t \log P(x_t
T5	Encoder-Decoder	Span Corruption	Text-to-text mapping with span masking

---

🛠️ Fine-Tuning Process Steps

1. **Load Pretrained Model and Tokenizer**
2. **Preprocess Input (Tokenize, Pad, Truncate)**
3. **Attach Task Head (Classification, LM, etc.)**
4. **Define Loss + Optimizer**
5. **Train on Downstream Task**

📌 Tips for Fine-Tuning

Tip	Explanation
Freeze Layers	Freeze some layers for smaller datasets to avoid overfitting
Use Learning Rate Scheduler	Transformers are sensitive to LR changes
Gradual Unfreezing	Start with top layers, unfreeze gradually

✅ Summary

Model	Type	Use Case	Pretraining Task
BERT	Encoder	Classification, QA	MLM + NSP

Model	Type	Use Case	Pretraining Task
GPT	Decoder	Text Generation	Causal LM
T5	Encoder-Decoder	Translation, Summarization	Span Prediction

---