Rathinam Trainers

Curriculum: Generative AI

Rathinam Trainers

1. Introduction to AI and Machine Learning

- **Basics of Artificial Intelligence**
  - Definition and history of AI
  - Key concepts and terminologies
- **Overview of Machine Learning**
  - Types of machine learning: supervised, unsupervised, and reinforcement
  - Introduction to neural networks

2. Fundamentals of Generative AI

- **Types of Generative Models**
  - Autoencoders
  - Generative Adversarial Networks (GANs)
  - Variational Autoencoders (VAEs)
- **Basic Applications**
  - Image generation
  - Text generation
  - Data augmentation

3. Neural Networks

3.1. Introduction to Neural Networks

- **Fundamentals of Neural Networks**
  - History and evolution of neural networks.
  - Basic concepts: neurons, weights, biases, and activation functions.
  - How neural networks mimic human brain functions.
- **Building Simple Neural Networks**
  - Designing a simple perceptron.
  - Implementing basic neural networks in Python using TensorFlow or PyTorch.
  - Understanding the structure of layers: input, hidden, and output.

3.2. Training Neural Networks

- **Forward Propagation and Backpropagation**
  - Detailed exploration of how data moves through a network (forward propagation).
  - The role of backpropagation and gradient descent in training neural networks.
  - Loss functions and their importance in model optimization.
- **Optimization and Regularization**
  - Different optimization algorithms: SGD, Adam, RMSprop.
  - Techniques for regularization: Dropout, L1/L2 regularization.
  - Overfitting, underfitting, and how to manage them.

3.3. Deep Neural Networks

- **Architecture of Deep Neural Networks**
  - Advancing from simple to deep networks: why deeper can be better.
  - Activation functions revisited: ReLU, Softmax, and their variants.
  - Design principles for deep neural networks.

Rathinam Trainers

- **Practical Implementation of Deep Networks**
  - Hands-on coding session building a deep neural network.
  - Case studies: Image recognition and natural language processing.

3.4. Convolutional Neural Networks (CNNs)

- **Introduction to CNNs**
  - Why CNNs for image processing?
  - Understanding convolution operations, pooling layers, and filters.
  - Building a basic CNN model.
- **Advanced CNN Architectures**
  - Exploring famous architectures: AlexNet, VGG, ResNet.
  - Implementing a CNN for a complex image classification task.
  - Strategies to improve CNN performance and efficiency.

3.5. Recurrent Neural Networks (RNNs) and Variants

- **Basics of RNNs**
  - How RNNs work and their applications.
  - Problems with traditional RNNs: vanishing and exploding gradients.
  - Practical implementations of RNNs in sequence modeling tasks.
- **Advanced RNNs - LSTM and GRU**
  - Introduction to Long Short-Term Memory (LSTM) units and Gated Recurrent Units (GRU).
  - When to use LSTM over GRU and vice versa.
  - Applications in time series analysis, speech recognition, and more.

3.6. Labs

Lab 1: Implementing a Basic Neural Network

Objective: Build and train a simple neural network to solve a basic classification problem.

- **Tasks:**
  - Set up a Python environment with necessary libraries (TensorFlow or PyTorch).
  - Use a simple dataset like the Iris dataset for classification.
  - Build a neural network with an input layer, one hidden layer, and an output layer.
  - Train the model and evaluate its performance on a test set.

Lab 2: Exploring Backpropagation and Optimization

Objective: Understand the dynamics of backpropagation and experiment with different optimization algorithms.

- **Tasks:**
  - Implement a neural network on the MNIST dataset.
  - Apply backpropagation manually to understand the gradient descent process.
  - Experiment with different optimizers (SGD, Adam, RMSprop) and compare their impact on training speed and model accuracy.

Lab 3: Constructing a Deep Neural Network

Objective: Build a deep neural network to handle a more complex problem.

Rahinam Trainers

- **Tasks:**
    - Choose a complex dataset, such as CIFAR-10.
    - Design a deep neural network with multiple hidden layers.
    - Implement activation functions such as ReLU and Softmax in different layers.
    - Train the deep neural network and analyze the improvements in learning and performance.

Lab 4: Developing a Convolutional Neural Network (CNN)

Objective: Create a CNN to perform image classification.

- **Tasks:**
    - Understand the structure and function of convolutional layers and pooling layers.
    - Build a CNN to classify images from a dataset like CIFAR-10 or Fashion-MNIST.
    - Visualize the filters and feature maps in the convolutional layers.
    - Evaluate the performance and discuss the advantages of CNNs over fully connected networks for image tasks.

Lab 5: Implementing Recurrent Neural Networks (RNN)

Objective: Use RNNs for a sequence modeling task such as time series prediction or text generation.

- **Tasks:**
    - Explain the architecture of RNNs and the problem of vanishing gradients.
    - Build an RNN to predict the next word in a sentence using a part of a text corpus.
    - Upgrade the RNN to LSTM or GRU to improve the handling of long-term dependencies.
    - Assess the performance improvements with LSTM/GRU over basic RNNs.

Lab 6: Capstone Project

Objective: Apply the skills learned to design a neural network for a real-world application.

- **Tasks:**
    - Identify a real-world problem suitable for a neural network solution. Examples might include speech recognition, anomaly detection in network traffic, or automated driving systems.
    - Collect and preprocess the data needed for the project.
    - Design, implement, and train a neural network model, utilizing appropriate architectures learned through the course (CNN, RNN, LSTM, etc.).
    - Present the project, including the problem statement, methodology, results, and future work suggestions.

4. Autoencoders

4.1. Introduction to Autoencoders

- **What are Autoencoders?**
    - Definition and fundamental concepts
    - Overview of the architecture: encoder and decoder
    - Applications and importance in machine learning
- **Mathematics Behind Autoencoders**

Rathinam Trainers

- Loss functions (mean squared error, binary cross-entropy)
- Optimization algorithms
- Underfitting vs. overfitting in autoencoders

4.2. Types of Autoencoders

- **Vanilla Autoencoders**
  - Structure and characteristics
  - Implementation challenges
- **Convolutional Autoencoders**
  - Advantages of using convolutional layers
  - Applications in image processing
- **Variational Autoencoders (VAEs)**
  - Introduction to probabilistic encoders and decoders
  - The reparameterization trick
  - Generating new data instances

4.3. Practical Applications

- **Autoencoders in Dimensionality Reduction**
  - Comparison with PCA
  - Visualizing high-dimensional data
- **Autoencoders for Denoising**
  - Building a denoising autoencoder
  - Practical exercises: Image and signal denoising
- **Anomaly Detection with Autoencoders**
  - Identifying anomalies in time-series data and images
  - Case study: Fraud detection in financial transactions

4.4. Labs

Lab 1: Building a Basic Autoencoder

Objective: Learn to construct and train a simple autoencoder using TensorFlow or PyTorch.

- **Tasks:**
  - Set up the programming environment (IDE, necessary libraries).
  - Create a simple autoencoder model for dimensionality reduction.
  - Use the MNIST dataset to train the model.
  - Visualize the encoded representations and the reconstructions.

Lab 2: Implementing a Convolutional Autoencoder

Objective: Develop a convolutional autoencoder to work with image data.

- **Tasks:**
  - Understand the difference in architecture between fully connected and convolutional autoencoders.
  - Build a convolutional autoencoder using the CIFAR-10 dataset.
  - Train the model to denoise images.
  - Evaluate the performance by comparing original, noisy, and denoised images.

Lab 3: Variational Autoencoder (VAE)

Rathinam Trainers

Objective: Build a Variational Autoencoder and explore its ability to generate new images.

- **Tasks:**
  - Review the theoretical basis of VAEs and the reparameterization trick.
  - Construct and train a VAE on a dataset like Fashion MNIST.
  - Generate new items by sampling from the latent space.
  - Discuss the differences in output between a basic autoencoder and a VAE.

Lab 4: Advanced Autoencoder Applications

Objective: Apply autoencoders to real-world scenarios such as anomaly detection or feature extraction.

- **Tasks:**
  - Choose a dataset appropriate for anomaly detection (e.g., credit card transactions, network traffic).
  - Build an autoencoder that learns to encode normal operations.
  - Detect anomalies by measuring the reconstruction loss.
  - Extract features using an autoencoder and use those features for a classification task.

Lab 5: Capstone Project

Objective: Utilize the skills learned to tackle a complex problem with autoencoders.

- **Tasks:**
  - Identify a problem that can be addressed with autoencoders, such as data compression, unsupervised clustering, or complex denoising tasks.
  - Design, implement, and train an autoencoder solution.
  - Present the project, detailing the approach, results, and insights gained.

5. Generative Adversarial Networks (GANs)

5.1. Introduction to GANs

- **Foundations of GANs**
  - History and development of GANs
  - Key concepts: Generator, Discriminator, and Adversarial process
  - Overview of the GAN architecture and how it works
- **The Mathematics Behind GANs**
  - Loss functions and training dynamics
  - Convergence theory
  - Understanding mode collapse and convergence issues

5.2. Training GANs

- **Setting Up for GAN Training**
  - Choosing the right hardware and software
  - Data preprocessing and augmentation for GANs
- **Session 2: GAN Training Techniques**
  - Techniques to stabilize training (BatchNorm, learning rate schedules)
  - Tips for monitoring GAN training (checking for mode collapse, using TensorBoard)

Rathinam Trainers

5.3. Advanced GAN Architectures

- **Variants of GANs**
  - Conditional GANs, DCGANs, and Wasserstein GANs
  - Applications and advantages of each variant
- **Innovative GAN Models**
  - Progressive growing of GANs
  - BigGAN, StyleGAN, and CycleGAN

5.4. Applications of GANs

- **Practical Applications**
  - Image synthesis and creative AI
  - Super-resolution, style transfer, and photo editing
- **Beyond Images**
  - GANs for generating text, music, and video
  - Ethical implications and potential misuse of GAN technology

Rathinam Trainers

5.5. Labs

Lab 1: Introduction to GANs

Objective: Build a simple GAN to generate digits similar to those in the MNIST dataset.

- **Tasks:**
  - Set up the programming environment.
  - Construct the basic architecture of a GAN, including both the generator and discriminator.
  - Train the GAN on the MNIST dataset.
  - Evaluate the generator by visualizing the quality of the generated digits.

Lab 2: Deep Convolutional GAN (DCGAN)

Objective: Implement a DCGAN to generate higher quality images.

- **Tasks:**
  - Understand the modifications to the basic GAN architecture to accommodate convolutional layers.
  - Build and train a DCGAN on the CIFAR-10 dataset.
  - Observe how the inclusion of convolutional layers improves image quality.
  - Discuss the stability of training and explore methods to optimize training.

Lab 3: Conditional GANs (cGANs)

Objective: Explore how conditional GANs can be used to generate images based on labels or conditions.

- **Tasks:**
  - Modify a GAN to accept additional labels as input to control the generation process.
  - Train a cGAN on a dataset with labeled data (e.g., Fashion MNIST).
  - Generate images conditioned on specific labels and analyze the outputs.

Lab 4: CycleGAN for Image-to-Image Translation

Objective: Implement a CycleGAN for tasks that require image translation without paired examples.

- **Tasks:**
  - Understand the concept of cycle consistency loss.
  - Build and train a CycleGAN to convert horses to zebras using unpaired images from different datasets.
  - Evaluate the effectiveness of the model in maintaining key attributes between the source and target domains.

Lab 5: StyleGAN for High-Quality Face Generation

Objective: Utilize StyleGAN to generate high-resolution, photorealistic images of human faces.

- **Tasks:**
  - Explore the architecture changes and techniques introduced in StyleGAN.
  - Implement and train a simplified version of StyleGAN using a dataset like CelebA.
  - Generate diverse, high-quality faces and explore the control over specific features through the style mixing technique.

Lab 6: Capstone Project

Objective: Apply GANs to a creative or innovative application of the student's choosing.

- **Tasks:**
  - Identify a unique problem or opportunity for applying GANs.
  - Design, implement, and refine a GAN solution, possibly integrating elements from previous labs.
  - Document the process, challenges, and results.
  - Present the project to peers and instructors for feedback and critique.

6. Variational Autoencoders (VAEs)

6.1. Introduction to VAEs

- **Understanding VAEs**
  - Overview of autoencoders and the introduction of VAEs
  - Theoretical underpinnings: from autoencoders to variational inference
  - Differences between standard autoencoders and VAEs
- **Components of VAEs**
  - Deep dive into the encoder, decoder, and loss function components
  - Introduction to the reparameterization trick and its necessity
  - Key mathematical concepts: KL divergence, probability distributions

6.2. Building and Training VAEs

- **Setting up Your Environment**
  - Tools and libraries required (TensorFlow, PyTorch)
  - Preparing datasets (e.g., MNIST, CIFAR-10) for experiments
- **Implementing a Basic VAE**
  - Step-by-step coding session to build a VAE
  - Training the VAE on a simple dataset like MNIST

Rathinam Trainers

- Analyzing the results: latent space, reconstruction quality

6.3. Advanced VAE Concepts

- **Variational Techniques and Optimizations**
  - Advanced variational techniques to improve VAEs
  - Addressing common issues: posterior collapse, imbalanced KL divergence
- **Extensions of VAEs**
  - Exploring different VAE architectures: Conditional VAEs, Hierarchical VAEs
  - Application-specific adaptations (e.g., for images, text, and audio)

6.4. Labs

Lab 1: Introduction to VAEs

Objective: Understand the basic components and functionality of VAEs.

- **Tasks:**
  - Setup the programming environment with all necessary libraries.
  - Construct a simple VAE model to learn the fundamentals of the encoder, decoder, and loss functions.
  - Train the VAE on the MNIST dataset to generate hand-written digits.
  - Visualize and analyze the latent space representation and the reconstruction quality.

Lab 2: Exploring the Reparameterization Trick

Objective: Dive deeper into the reparameterization trick, a key component of VAEs that allows backpropagation.

- **Tasks:**
  - Discuss the mathematical theory behind the reparameterization trick.
  - Implement the trick within a VAE model.
  - Observe how the model performance changes with and without the use of the trick.

Lab 3: Advanced VAE Architectures

Objective: Implement and compare different VAE architectures.

- **Tasks:**
  - Build a Conditional VAE (CVAE) to generate data based on conditional inputs.
  - Explore a Hierarchical VAE (HVAE) to understand how complex data structures can be modeled.
  - Compare the performance, benefits, and drawbacks of each architecture.

Lab 4: VAEs for Image Generation

Objective: Use VAEs to generate complex images.

- **Tasks:**
  - Train a VAE on a more complex dataset such as CelebA or CIFAR-10.
  - Experiment with different network architectures to improve image quality.
  - Analyze the generated images for diversity and realism.

Rathinam Trainers

Lab 5: Anomaly Detection with VAEs

Objective: Apply VAEs to identify anomalies in dataset.

- **Tasks:**
  - Choose a dataset suitable for anomaly detection (e.g., credit card transaction data).
  - Train a VAE model to encode normal patterns.
  - Detect anomalies by measuring the reconstruction error: higher errors indicate potential anomalies.

Lab 6: Capstone Project

Objective: Utilize the skills learned to tackle a complex problem using VAEs.

- **Tasks:**
  - Identify a unique challenge that can be addressed with VAEs, such as synthesizing music, designing new molecules, or creating art.
  - Design and implement a VAE to address the chosen challenge.
  - Present the project, highlighting the approach, results, and insights gained.

7. Fine Tuning

7.1. Introduction to Model Fine-Tuning

- **Basics of Model Fine-Tuning**
  - Overview of model fine-tuning and transfer learning concepts.
  - Differences between fine-tuning, transfer learning, and from-scratch training.
  - Understanding when and why to fine-tune.
- **Preparing for Fine-Tuning**
  - Selecting the right pre-trained models.
  - Data requirements: Understanding how much data is needed.

7.2. Techniques in Fine-Tuning

- **Fine-Tuning Strategies**
  - How to fine-tune: frozen layers vs. trainable layers.
  - Learning rate adjustments and scheduler options.
  - Regularization techniques to prevent overfitting during fine-tuning.
- **Fine-Tuning in Practice**
  - Step-by-step guide to fine-tuning a convolutional neural network (CNN) for image classification.
  - Fine-tuning a natural language processing (NLP) model using BERT for sentiment analysis.

7.3. Advanced Topics in Fine-Tuning

- **Fine-Tuning for Large Datasets**
  - Strategies for handling large-scale data in fine-tuning.
  - Utilizing mixed-precision training and distributed training techniques.
  - Case studies: Fine-tuning models on high-resolution images or large text corpora.
- **Hyperparameter Optimization**
  - Techniques for optimizing hyperparameters during fine-tuning.

Rathinam Trainers

- Tools for automatic hyperparameter tuning: Grid Search, Random Search, Bayesian Optimization.

7.4. Labs

Lab 1: Fine-Tuning Basics with a Pre-trained Model

Objective: Learn the basic steps of fine-tuning a pre-trained model for a new task.

- **Tasks:**
  - Set up the programming environment and install necessary libraries (TensorFlow or PyTorch).
  - Load a pre-trained model (e.g., ResNet for images, BERT for text).
  - Fine-tune the model on a small new dataset (e.g., a subset of CIFAR-10 for image classification or a small sentiment analysis dataset for text).
  - Evaluate the performance before and after fine-tuning to understand the impact.

Lab 2: Advanced Fine-Tuning Techniques

Objective: Explore advanced strategies for effective fine-tuning.

- **Tasks:**
  - Implement various learning rate schedules (e.g., cyclic learning rates, exponential decay).
  - Experiment with different layers of the model frozen versus trainable.
  - Use data augmentation techniques to improve model robustness and prevent overfitting.

Lab 3: Fine-Tuning for Small Data Sets

Objective: Master the technique of fine-tuning pre-trained models on small datasets.

- **Tasks:**
  - Choose a pre-trained model and a small dataset from a different domain (e.g., medical images, rare text corpus).
  - Apply techniques like few-shot learning or data augmentation to maximize learning from limited data.
  - Assess model performance and adjust fine-tuning parameters to optimize results.

Lab 4: Multi-Task Learning

Objective: Implement fine-tuning for multi-task learning, where a single model learns to perform multiple tasks.

- **Tasks:**
  - Load a model that can be adapted for multiple tasks (e.g., a transformer model for different NLP tasks).
  - Set up the model to share common features while fine-tuning task-specific layers.
  - Train and evaluate the model on multiple tasks simultaneously and analyze the trade-offs.

Lab 5: Domain Adaptation

Objective: Learn to adapt a model from one domain to another significantly different domain.

Rahinam Trainers

- **Tasks:**
    - Select a model trained on one type of data and fine-tune it for a related but different type of data (e.g., adapting a model from recognizing everyday objects to medical diagnostic images).
    - Implement domain adaptation techniques to minimize domain shift issues.
    - Evaluate the effectiveness of the adapted model on the new domain.

7.5. Capstone Project and Industry Applications

- **Planning Your Fine-Tuning Project**
    - Identifying a problem that can be addressed through fine-tuning.
    - Dataset selection and preprocessing.
    - Initial model choice and setup.
- **Implementing and Optimizing Your Project**
    - Practical application of fine-tuning techniques.
    - Continuous monitoring and iterative improvements.
- **Project Presentation and Review**
    - Presentation of the projects to peers and instructors.
    - Review and feedback session to discuss challenges, solutions, and insights.

8. Transfer Learning

Transfer Learning provides a pathway for learners to harness pre-trained models and adapt them to new, often quite different tasks, effectively and efficiently.

8.1. Foundations of Transfer Learning

- **Introduction to Transfer Learning**
    - Definitions and key concepts in transfer learning.
    - The importance and benefits of using transfer learning in various domains.
    - Overview of scenarios where transfer learning is applicable.
- **Understanding Source and Target Tasks**
    - The concept of domain adaptation.
    - Criteria for selecting source models.
    - Differences between similar and dissimilar domain transfer learning.

8.2. Implementing Transfer Learning

- **Tools and Frameworks**
    - Introduction to popular tools (TensorFlow, PyTorch, Keras, Hugging Face).
    - Hands-on: Loading pre-trained models and modifying them for new tasks.
- **Practical Application in Image Processing**
    - Step-by-step guide to adapting a pre-trained image recognition model to a new task.
    - Techniques to handle different dataset sizes and feature spaces.

8.3. Transfer Learning in NLP and Beyond

- **Session 1: NLP Applications**
    - Adapting BERT-like models for tasks like sentiment analysis, question answering, and more.
    - Discussion on the nuances of language model adaptation.

- **Session 2: Cross-Domain Transfer Learning**
  - Strategies for transferring knowledge across vastly different data types and tasks.
  - Case studies in areas like healthcare, finance, and autonomous vehicles.

8.4. Challenges and Ethical Considerations

- **Challenges in Transfer Learning**
  - Addressing common pitfalls: negative transfer, overfitting on small target datasets.
  - Methods to evaluate transfer learning effectiveness.
- **Ethics and Future of Transfer Learning**
  - Ethical considerations in using transfer learning.
  - Predictions for future trends and innovations in transfer learning.

8.5. Labs

Lab 1: Introduction to Transfer Learning

Objective: Learn the fundamentals of transfer learning and apply a pre-trained model to a new task.

- **Tasks:**
  - Setup the programming environment and familiarize with necessary libraries.
  - Load a pre-trained image classification model (like VGG16 or ResNet) using TensorFlow or PyTorch.
  - Fine-tune the model on a new dataset (e.g., a specific type of animal or plant classification).
  - Evaluate performance improvements and understand the impact of transfer learning.

Lab 2: Customizing Models for Transfer Learning

Objective: Adapt a pre-trained model's architecture to better suit a specific task.

- **Tasks:**
  - Modify the top layers of a pre-trained model to fit a new classification problem.
  - Experiment with different architectures and numbers of layers to optimize performance.
  - Train and validate the model on a new dataset, focusing on tuning hyperparameters like learning rate and batch size.

Lab 3: Transfer Learning with NLP Models

Objective: Apply transfer learning to natural language processing using a pre-trained BERT model.

- **Tasks:**
  - Load a pre-trained BERT model and adapt it for a sentiment analysis task.
  - Prepare and preprocess text data for training.
  - Fine-tune the model on a dataset containing movie reviews or customer feedback.
  - Analyze the performance and discuss the benefits of using transfer learning for NLP.

Lab 4: Cross-Domain Transfer Learning

Rathinam Trainers

Objective: Implement transfer learning across different domains or data types.

- **Tasks:**
    - Choose two significantly different datasets (e.g., natural images vs. medical images).
    - Adapt a model trained on the first dataset to perform tasks on the second dataset.
    - Implement and evaluate domain adaptation techniques to minimize performance loss.
    - Discuss challenges encountered and strategies to overcome them.

Lab 5: Advanced Transfer Learning Techniques

Objective: Explore advanced strategies and techniques in transfer learning.

- **Tasks:**
    - Implement and compare various feature extraction and fine-tuning strategies.
    - Use techniques such as progressive freezing of layers or gradual unfreezing.
    - Apply and evaluate different learning rate schedules to optimize transfer learning results.

8.6. Capstone Project

- **Planning and Implementation**
    - Identify a unique problem suitable for a transfer learning approach.
    - Select appropriate datasets and pre-trained models.
    - Implement, test, and refine the model.
- **Presentation and Review**
    - Present the final project to peers and instructors.
    - Engage in a critical review session to discuss each project's strategy, implementation, and outcomes.

9. Transformer Models

9.1. Introduction to Transformer Models

- **Background and Motivation**
    - Historical context and the evolution of neural networks.
    - Limitations of RNNs and LSTMs that led to the development of transformers.
    - Overview of the "Attention is All You Need" paper.
- **Core Concepts**
    - Understanding attention mechanisms.
    - Self-attention vs. traditional attention mechanisms.
    - Key components: embeddings, positional encoding, and multi-head attention.

9.2. Transformer Architecture

- **Encoder-Decoder Structure**
    - Detailed breakdown of the encoder and decoder architecture.
    - Role of each component: multi-head attention, feed-forward networks, and normalization.
    - Visualizing the flow of data through the transformer.
- **Implementation Basics**
    - Building a simple transformer model from scratch using TensorFlow or PyTorch.

Rathinam Trainers

- Implementing key components: attention layers, positional encodings, and feed-forward networks.

9.3. Training Transformers

- **Data Preparation and Preprocessing**
  - Preparing text data for training.
  - Tokenization and encoding strategies.
  - Creating input and output sequences for the transformer model.
- **Training Techniques**
  - Loss functions and optimization for transformer models.
  - Understanding and implementing learning rate schedules.
  - Techniques for efficient training, such as gradient clipping and mixed precision training.

9.4. Applications of Transformers

- **Natural Language Processing Tasks**
  - Text classification, sentiment analysis, and named entity recognition.
  - Machine translation and text summarization.
  - Question answering and conversational AI.
- **Hands-On Projects**
  - Implementing a transformer for a specific NLP task (e.g., translation or summarization).
  - Fine-tuning pre-trained models like BERT, GPT-2, or T5 on custom datasets.
  - Evaluating model performance and interpreting results.

9.5. Advanced Topics and Optimization

- **Transformer Variants**
  - Exploring BERT, GPT, T5, and other transformer-based models.
  - Differences and specific use cases for each variant.
  - Practical applications and case studies.
- **Model Optimization and Scaling**
  - Techniques for scaling transformers for large datasets.
  - Distributed training and multi-GPU setups.
  - Fine-tuning and hyperparameter optimization.

9.6. Labs

Lab 1: Introduction to Transformer Models

Objective: Understand the basics of transformers and implement key components.

- **Tasks:**
  - Set up the Python environment with necessary libraries (TensorFlow or PyTorch).
  - Implement a simple self-attention mechanism.
  - Create positional encodings and visualize them.
  - Build a multi-head attention layer and test it with sample input data.

Lab 2: Building the Transformer Architecture

Objective: Construct the full transformer architecture and understand its data flow.

Rahman Trainers

- **Tasks:**
    - Implement the encoder layer: multi-head attention and feed-forward network.
    - Implement the decoder layer: masked multi-head attention, encoder-decoder attention, and feed-forward network.
    - Combine encoder and decoder layers to build a complete transformer model.
    - Test the transformer model with a small dataset.

Lab 3: Training a Transformer Model

Objective: Train a transformer model on a specific task.

- **Tasks:**
    - Prepare a dataset (e.g., English-French translation dataset).
    - Implement tokenization and sequence encoding.
    - Set up the training loop with appropriate loss function and optimizer.
    - Train the transformer model and monitor training progress with evaluation metrics.

Lab 4: Fine-Tuning Pre-trained Transformers

Objective: Fine-tune a pre-trained transformer model on a custom dataset.

- **Tasks:**
    - Load a pre-trained model like BERT or GPT-2.
    - Prepare a custom dataset for a specific NLP task (e.g., sentiment analysis).
    - Fine-tune the pre-trained model on the custom dataset.
    - Evaluate the performance of the fine-tuned model and visualize results.

Lab 5: Advanced Applications of Transformers

Objective: Apply transformers to various advanced NLP tasks.

- **Tasks:**
    - Implement a transformer model for text summarization using a dataset like CNN/Daily Mail.
    - Train a transformer for question answering using the SQuAD dataset.
    - Explore text generation with a pre-trained GPT model.
    - Analyze the results and compare the performance on different tasks.

Lab 6: Capstone Project

Objective: Apply the knowledge and skills learned to solve a real-world problem using transformers.

- **Tasks:**
    - Identify a real-world problem suitable for transformer-based solutions (e.g., language translation, chatbot development, text summarization).
    - Design a project plan, including data collection, preprocessing, model selection, and implementation.
    - Implement and train the transformer model.
    - Evaluate and refine the model, then present the project results.

Ratinam Trainers