30 Days of PyTorch

Day 1: Installation and "Hello, PyTorch!" - Install PyTorch and run a basic PyTorch program to print "Hello, PyTorch!" to get started.

Day 2: Tensor Operations - Learn and perform basic tensor operations like addition, subtraction, multiplication, and division using PyTorch.

Day 3: Linear Regression - Implement a simple linear regression model using PyTorch for predicting continuous values.

Day 4: Convolutional Neural Networks (CNN) - Build a CNN using PyTorch for image classification tasks.

Day 5: Activation Functions - Explore different activation functions such as ReLU, sigmoid, and tanh, and understand their impact on model performance.

Day 6: Model Training and Evaluation - Train a neural network on a small dataset, evaluate its accuracy, and analyze the results.

Day 7: Recurrent Neural Networks (RNN) - Implement an RNN using PyTorch for sequence prediction tasks.

Day 8: Loss Functions - Experiment with different loss functions like mean squared error or cross-entropy and observe their effects on model training.

Day 9: Transfer Learning - Utilize pre-trained models from PyTorch's model zoo for transfer learning and fine-tune them on a new dataset.

Day 10: Optimization Algorithms - Implement and compare different optimization algorithms like SGD, Adam, or RMSprop for model training.

Day 11: Generative Adversarial Networks (GAN) - Implement a GAN using PyTorch for generating synthetic images.

Day 12: Automatic Differentiation - Understand PyTorch's automatic differentiation mechanism (autograd) and manually compute gradients.

Day 13: Reinforcement Learning - Build a deep reinforcement learning agent using PyTorch for a simple environment.

Day 14: Data Loading and Preprocessing - Explore PyTorch's data loading utilities and preprocess image data, including data augmentation.

Day 15: Unsupervised Learning with VAE - Implement a variational autoencoder (VAE) using PyTorch for unsupervised learning tasks.

Day 16: Hyperparameter Tuning - Experiment with different network architectures and hyperparameters to optimize model performance.

Day 17: GPU Acceleration - Utilize PyTorch's GPU acceleration capabilities and train models on a GPU for faster computation.

Day 18: Fine-tuning and Transfer Learning - Fine-tune a pre-trained PyTorch model on a specific task or dataset for improved performance.

Day 19: Visualization with TensorBoard - Integrate PyTorch with TensorBoard to visualize training metrics and monitor model performance.

Day 20: Sequence-to-Sequence Models - Implement a sequence-to-sequence (Seq2Seq) model using PyTorch for machine translation tasks.

Day 21: Distributed Training - Explore PyTorch's distributed training capabilities for scaling up model training across multiple machines.

Day 22: Regularization Techniques - Experiment with different regularization techniques like dropout or L1/L2 regularization to improve model generalization.

Day 23: Custom Components - Implement a custom loss function or a custom layer in PyTorch to extend model capabilities.

Day 24: Production-Ready Models with TorchScript - Use PyTorch's torchscript to convert a PyTorch model to a production-ready format.

Day 25: Model Interpretability - Explore PyTorch's model interpretability tools, such as saliency maps or feature visualization.

Day 26: Optimization Strategies - Optimize model training with techniques like learning rate scheduling, early stopping, or gradient clipping.

Day 27: Natural Language Processing with Transformers - Implement a transformer model using PyTorch for NLP

Day 28: Model Deployment - Learn about deploying PyTorch models in production environments, including techniques like model serialization and serving via REST APIs.

Day 29: Model Compression - Explore techniques for model compression and optimization, such as pruning, quantization, or knowledge distillation, to reduce model size or improve inference speed.

Day 30: Reflection and Showcase - Reflect on your PyTorch learning journey, showcase your favorite project or experiment, and plan for future explorations in deep learning and PyTorch.