**DL Types / Architectures**

* Feedforward Neural Network (FNN) - Standard fully connected network (Dense layers, input -->hidden -->output)
* Convolutional Neural Network (CNN) - Specialized for spatial data (images) (Conv layers, Pooling layers, Flatten, FC layers)
* Recurrent Neural Network (RNN) - For sequential / time-series data (Vanilla RNN, LSTM, GRU)
* Transformer - Attention-based sequence modelling (Encoder-Decoder, Self-Attention, BERT, GPT)
* Autoencoder - Unsupervised representation learning (Encoder -->latent space -->Decoder)
* Generative Adversarial Network (GAN) - Generates synthetic data (Generator + Discriminator, adversarial training)

**Layers / Components**

* Dense / Fully Connected - Standard linear transformation y = Wx + b (Dense layers)
* Convolutional - Extract spatial features (Kernel/Filter sliding over input)
* Pooling - Downsample features (Max pooling, Average pooling)
* Dropout - Regularization by randomly dropping neurons (Dropout rate e.g., 0.2)
* Batch Normalization - Stabilize training by normalizing layer outputs (BatchNorm layers)
* Activation Functions - Introduce non-linearity (ReLU, Sigmoid, Tanh, Softmax, GELU)

**Loss / Objective Functions**

* Regression - Measure continuous prediction error (MSE, MAE, Huber Loss)
* Classification - Measure misclassification (Cross-Entropy, Categorical CE, Hinge Loss)
* Custom Loss - Task-specific objectives (Dice Loss - segmentation, Perplexity - NLP)

**Optimizers / Training**

* Gradient Descent Variants - Update model parameters (SGD, SGD with Momentum, Adam, RMSProp, Adagrad)
* Learning Rate Scheduling - Adjust LR during training (Step decay, Cosine annealing, ReduceLRonPlateau)

**Regularization / Overfitting Prevention**

* Techniques - Improve generalization (L1/L2 weight decay, Dropout, Early Stopping, Data Augmentation)

**Evaluation Metrics**

* Regression Metrics - Evaluate continuous outputs (RMSE, MAE, R²)
* Classification Metrics - Evaluate discrete outputs (Accuracy, Precision, Recall, F1, ROC-AUC)
* Specialized Metrics - Task-specific evaluation (IoU/mAP - CV, BLEU/ROUGE - NLP, Perplexity)

**Initialization / Tricks**

* Weight Initialization - Helps convergence (Xavier, He, Kaiming initialization)
* Gradient Clipping - Avoid exploding gradients in RNNs (Clip by norm or value)

**Computational Considerations**

* GPU / TPU Usage - Accelerate training for large networks (CUDA, cuDNN, Tensor Cores)
* Batch Size & Epochs - Trade-off between speed and stability (Small batch -->noisy gradient, Large batch -->stable but memory heavy)

**Notes / Tips**

* Transfer Learning - Reuse pre-trained models for new tasks (ResNet, BERT, GPT, EfficientNet)
* Fine-Tuning - Adjust pre-trained model weights (Freeze layers + train top layers)