In machine learning, particularly in neural networks, several algorithms and techniques are used to optimize performance. Here is a breakdown of key categories like activation functions, optimizers, loss functions, and more:

1. Activation Functions

- **Sigmoid (Logistic)**: `σ(x) = 1 / (1 + e^(-x))`
- **Tanh**: `tanh(x) = (e^x - e^(-x)) / (e^x + e^(-x))`
- **ReLU (Rectified Linear Unit)**: `ReLU(x) = max(0, x)`
- **Leaky ReLU**: `Leaky ReLU(x) = max(0.01x, x)`
- **ELU (Exponential Linear Unit)**: `ELU(x) = α(e^x - 1)` for `x < 0`, `x` otherwise
- **Softmax**: Often used in the output layer for multi-class classification: `Softmax(x_i) = e^(x_i) / Σ e^(x_j)`

2. Optimizers

- **SGD (Stochastic Gradient Descent)**: Updates weights based on individual samples.
- **Mini-Batch Gradient Descent**: A compromise between SGD and full-batch gradient descent.
- **Momentum**: Enhances SGD by adding a momentum term to the updates.
- **Adam (Adaptive Moment Estimation)**: Combines momentum and adaptive learning rates.
- **RMSprop**: Uses a moving average of squared gradients to normalize the gradients.
- **Adagrad**: Adaptively scales the learning rate based on past gradients.
- **Adadelta**: A variant of Adagrad with a decaying learning rate.
- **Nadam**: Adam with Nesterov momentum.

3. Loss Functions

- **Mean Squared Error (MSE)**: `MSE = (1/n) Σ (Y' - Y)^2` — Commonly used for regression tasks.
- **Mean Absolute Error (MAE)**: `MAE = (1/n) Σ |Y' - Y|`
- **Binary Cross-Entropy**: Used for binary classification: `-(1/n) Σ [Y log(Y') + (1 - Y) log(1 - Y')]`
- **Categorical Cross-Entropy**: Used for multi-class classification.
- **Huber Loss**: A combination of MSE and MAE to handle outliers more gracefully.
- **Hinge Loss**: Primarily used for "maximum-margin" classifiers like SVMs.
- **Kullback-Leibler Divergence (KL Divergence)**: Measures how one probability distribution diverges from a second, reference probability distribution.

4. Regularization Techniques

- **L1 Regularization (Lasso)**: Adds the sum of the absolute values of the coefficients to the loss function: `Loss + λ Σ |w_i|`
- **L2 Regularization (Ridge)**: Adds the sum of the squares of the coefficients to the loss function: `Loss + λ Σ w_i^2`
- **Dropout**: Randomly "drops" units (along with their connections) during training to prevent overfitting.
- **Batch Normalization**: Normalizes the inputs of each layer to stabilize and speed up training.

5. Initialization Techniques

- **Random Initialization**: Weights are initialized randomly.
- **Xavier (Glorot) Initialization**: Designed to keep the variance of the input and output of each layer the same.
- **He Initialization**: Similar to Xavier, but optimized for ReLU activations.
- **LeCun Initialization**: Designed for "scaled" versions of activations like Sigmoid.

6. Learning Rate Schedulers

- **Step Decay**: Reduces the learning rate by a factor every few epochs.
- **Exponential Decay**: Reduces the learning rate exponentially.
- **Cosine Annealing**: Gradually decreases the learning rate following a cosine function.
- **Cyclical Learning Rate**: The learning rate cyclically oscillates between a range of values.

7. Evaluation Metrics

- **Accuracy**: The ratio of correctly predicted instances to the total instances.
- **Precision**: The number of true positives divided by the number of true positives plus false positives.
- **Recall (Sensitivity)**: The number of true positives divided by the number of true positives plus false negatives.
- **F1 Score**: The harmonic mean of precision and recall.
- **AUC-ROC**: The area under the receiver operating characteristic curve; used for binary classification.
- **Confusion Matrix**: A table used to describe the performance of a classification model.

8. Dimensionality Reduction Techniques

- **Principal Component Analysis (PCA)**: Projects data onto lower dimensions.
- **t-SNE (t-distributed Stochastic Neighbor Embedding)**: Visualizes high-dimensional data in a lower dimension.
- **Linear Discriminant Analysis (LDA)**: Finds a linear combination of features that characterizes or separates classes.

9. Ensemble Learning Techniques

- **Bagging**: Combines the predictions of multiple models (e.g., Random Forest).
- **Boosting**: Sequentially builds models that correct errors made by previous models (e.g., AdaBoost, XGBoost, Gradient Boosting).
- **Stacking**: Combines multiple models using a meta-model to improve predictions.