Model Training

Model Operation

- Takes input data, performs matrix multiplication to generate output.
- **Training Phase**: Involves both input and output to train the model.
- **Deployment Phase**: Uses input and the trained model architecture to make predictions.
- **Inference Phase**: The stage where the model is used after deployment.

Training:

- **Iterative Improvement**: Process of refining model parameters to transform input into accurate output.
  - **Example**: Classification task for dog, cat, and duck images. Initially, the model guesses with around 33% accuracy (random guessing). Through training, it updates parameters to better identify each class.
- **Layer Learning**: Each layer in the model learns different parameters, represented numerically.
- Goal: Minimize errors on both training data (seen data) and new data (testing data).

Data Types:

- **Training Data**: Used to teach the model and update parameters.

- **Validation Data**: Periodically tests the model during training to check improvement.

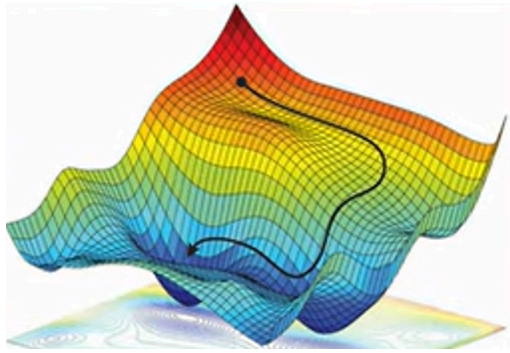
- **Test Data**: Tests the model's final accuracy after training.

- **Parameter Space**: Space containing all possible values for model parameters. Example: A model with 3 billion parameters starts with undefined values.

- **Optimal Point**: The spot in parameter space with the least error. The model aims to reach this point.

- **Loss Function**: Measures the difference between the model's predictions and actual values.

- **Gradient Descent**: Algorithm to adjust parameters in the direction that minimizes the loss function, moving toward the optimal point.



Overfitting and Regularization:

- **Overfitting**: When the model performs exceptionally well on training data but poorly on validation/test data.
    - **Regularization Techniques**:
        - **Dropout**: Randomly drops neurons during training to prevent dependency on certain neurons, promoting a more generalized learning.
        - **Early Stopping**: Stops training when the model starts overfitting, indicated by a significant performance gap between training and validation data.

Underfitting:

- **Underfitting**: When the model performs poorly on training, validation, and test data.
    - **Solutions**:
        - Add more layers or neurons.
        - Reduce regularization (e.g., avoid dropout).
        - Increase epochs to allow the model to learn from the data multiple times.

Data Augmentation and Class Imbalance:

- **Data Augmentation**: Modifies existing data (e.g., changing angles, colors) to create more diverse training samples.
- **Class Imbalance**: When one class has significantly more data than others, leading to biased predictions.
    - **Solutions**:
        - **Oversampling**: Increase data for minority classes through augmentation (risk of overfitting).
        - **Undersampling**: Reduce data from majority classes (risk of losing valuable information).

Transfer Learning:

- **Pretrained Models**: Most models are not trained from scratch but are fine-tuned using new data.
- **Fine Tuning**: Adapts a pretrained model to new, specific features by training it further with new data.

Hyperparameters:

- **Hyperparameter**: A parameter whose value is set before the machine learning process begins, such as learning rate, batch size, and number of epochs.

Batch Size - Number of samples processed before the model's internal parameters are updated.

- **Impact:** Large batch sizes lead to more stable and accurate estimates of the gradient, while smaller batch sizes can introduce noise but often result in faster convergence.

Number of Epochs

- **Impact:** Too few epochs can lead to underfitting, while too many can lead to overfitting.

Learning Rate - The rate at which the model weights are updated.

- **Impact:** A high learning rate can cause the model to converge too quickly to a suboptimal solution. A low learning rate might result in a very slow convergence or getting stuck in local minima.