**Task 2: Training the Model and Optimizing Convergence**

**Objective**

The primary objective of this task is to train the previously defined Artificial Neural Network (ANN) model on the customer churn dataset, ensuring that the model converges efficiently and avoids overfitting. This involves defining training parameters, monitoring model performance, and applying optimization techniques to improve generalization on unseen data.

**Steps Taken**

**1. Data Preparation for Training**

- **Train/Test Split:**
    - The dataset was divided into two parts:
        - **Training Set:** Used to train the model. This contains the bulk of the data and is responsible for teaching the model the relationships between the features and the target (churn/no churn).
        - **Test Set:** This set is kept separate and is used to evaluate the model's performance after training.
- **Why we chose this approach:**
    - The act of splitting the data into two parts (training and testing) helps us to check how well our model is going with unknown data points. This allows us to assess whether the model is overfitting (vs generalising well to new customers) as it sees entirely different data in testing.

**2. Defining the Model Training Process**

- **Training Configuration:**
    - **Epochs:**
        - The model was trained over **50 epochs**. Each epoch represents one complete pass through the entire training dataset.
        - **Why we chose this configuration:**
            - 50 epochs provide sufficient iterations for the model to learn the underlying patterns in the data without overfitting. Early stopping was applied to prevent unnecessary training if the model's performance did not improve.
    - **Batch Size:**
        - A batch size of **32** was used during training.
        - **Why we chose this configuration:**
            - The batch size specifies the number of samples per parameter update. 32 is a common choice as that size provides ample training speed along with fast model convergence.
- **Validation Data:**

    - The model was validated on the test dataset after each epoch. This provides real-time feedback on how well the model is performing on unseen data during the training process.
    - **Why we chose this configuration:**
        - Monitoring validation performance is crucial to detect overfitting early. If the model performs well on the training data but poorly on the validation data, it indicates overfitting.

**3. Optimizing Convergence and Avoiding Overfitting**

- **Early Stopping:**
    - **What it is:** Early stopping is a technique that monitors the model's validation performance and halts training if performance stops improving.
    - **Why we used this technique:**
        - Training can sometimes continue past the point where the model is improving on the training data but degrading on validation data. By stopping early, we ensure that the model does not overfit and waste computational resources.
    - **Implementation:**
        - The model was fit with early stopping, guided to stop training when validation performance did not improve over a run of 10 epochs and the best weights were restored.
        - **Why we chose this configuration:**
            - This allows the model to train long enough to potentially improve but stops it from training for too long if no improvement is observed. Restoring the best weights ensures that the model's best version is retained.
- **Checkpoints:**
    - **What it is:** Checkpoints were created to save the best model during training. This way, if training is interrupted or if early stopping occurs, the best model can be retrieved.
    - **Why we used this technique:**
        - This allows the model to be trained long enough for a potential improvement but not too long if no real performance gain is observed. In this sense, restoring the best weights is making sure that our model stays in its peak form.
    - **Implementation:**
        - A checkpoint was defined to monitor **validation loss** (val_loss), and the best model was saved as best_model.h5.
        - **Why we chose this configuration:**
            - This also saves the best model we observed on validation data, and it guarantees that if training is interrupted for some reason (overfitting early happens as well) later or a smaller network size gives better generalization performance.

**Model Training Process**

- **Training the ANN Model:**
    - The model was trained using the training dataset, with the target variable being the churn outcome (binary classification: churn/no churn).
    - The **Adam optimizer** was used to adjust the model's internal weights based on the gradients of the loss function (binary cross-entropy).
        - **Why we used Adam optimizer:**
            - Adam (Adaptive Moment Estimation) is a robust and widely used optimizer that combines advantages of two other optimizers AdaGrad, an optimization method from the first order gradient about minimizing loss function & RMSProp. Two-sided Material Similarity Maximization (TMSM) This technique is adaptive, efficient and requires minimal tuning whichis good for our use-case as it can be used in deep learning.
    - **Loss Function:**
        - At the time, binary cross-entropy was what a model minimized during training. While binary cross-entropy is the defacto loss function for most of our problems dealing with predictions and answers only having two possible classes.
        - **Why we chose binary cross-entropy:**
            - This loss function computes the probability distance of predicted probabilities from actual labels (Churn/No Churn). In other words, it punishes too much the difference between predicted probability and true class.

**4. Monitoring and Saving Results**

- **Loss and Accuracy Visualization:**
    - **Training and Validation Loss:** Created graph which depicts the training and validation loss per epoch. It is used to check the performance of model weather its doing good or overfitting.
    - **Why we generated this visualization:**
        - Arguments such as this plot_loss() function allow us to draw the loss over time, which lets us investigate visually what took place during training. Overfitting, Example: The validation loss starts increasing while the training loss decreases.
    - **Training and Validation Accuracy:** Similarly, a graph was created for the accuracy on the training and validation datasets.
    - **Why we generated this visualization:**
        - The AFO is an answer to how accurate the model can keep track of whether customer churn or not. At least a gap forming in training / test accuracies can indicate overfitting.
    - **Result:** The training and validation accuracy continued to rise during the epochs which was a positive indication. However, there was a little wobble in the validation accuracy which can be attributed to the training on a real-world dataset.

**Results Obtained**

- **Training Results:**
    - The model was successfully trained over **50 epochs** with a batch size of **32**. However, early stopping was triggered at epoch **20**, restoring the best model weights from that epoch.
    - The **final training loss** was lower than the validation loss, suggesting a good fit to the data with minimal overfitting.
- **Saved Files:**
    - **Best Model:** The best model was saved as best_model.h5 during training.
    - **Training History:** The training and validation history, including loss and accuracy over time, was saved to training_results.txt for further analysis.

**Challenges Faced**

- **Fluctuations in Validation Performance:**
    - During training, there were fluctuations in validation accuracy. This could be due to noise in the dataset or differences in the characteristics between the training and validation sets. Although the early stopping mechanism helped, further data cleaning or more advanced regularization techniques could be explored to stabilize the performance.
- **Possible Overfitting:**
    - We observed a slight gap between training and validation accuracy, which indicates some overfitting. To mitigate this, techniques like dropout were applied, and early stopping was used to prevent further overfitting.

**Conclusion**

The training of the ANN model was successful since we had good data validation and good accuracy. Both the early stopping technique and the checkpointing techniques were salient in that they saved the best model while preventing the overfitting process. The Adam optimiser and the binary cross-entropy loss for the function were quite efficient since it maintained the accuracy high. Therefore, we have successfully trained the model and now the next step is to use the model to make predictions and evaluate the performance according to this implementation.