1 Implementation Process - 85.12%

Project Introduction: Implemented a Switch Transformer model for text classification on the IMDB sentiment dataset. This is the Structure of the code:

1.1 Data Augmentation & Embedding

- **random_deletion(), random_swap(), random_replacement()**: Randomly delete or replace some tokens with a predefined probability. Randomly swap the positions of n pairs of words.
- **apply_augmentation()**: Combined the three augmentation methods to enhance the input text.
- **TokenAndPositionEmbedding()**: Token and position embedding with random Gaussian noise injection:

$$x = x + \mathcal{N}(0, \text{noise\_std})$$

1.2 Router Module of Mixture of Experts (MoE) & Transformer Layers

- **Router Auxiliary Loss in load_balanced_loss(router_probs, expert_mask)**: The auxiliary loss for the router is designed to balance the load across experts. Try to make the distribution of each expert's actual load (density_1) and the mean routing probability (density_1_proxy) as close to a uniform distribution as possible by this loss function:

$$L_{aux} = E[density_1] \cdot E[density_1^{proxy}] \cdot (num\_experts)^2$$

- **Expert Network in ResNetFeedForward()**: This structure helps in vanishing gradients and significantly increases the model's capacity. Each expert in the MoE is built using a ResNetFeedForward block:

$$y = \text{ReLU}\left(W_2 \cdot \text{ReLU}\left(W_1 \cdot x\right)\right) + x$$

- **Mixture of Experts in Route() & Switch()**: router_logits is the score of each expert in the token and then softmax it to router_probs. Get top-1 expert per token. Compute cumulative positions for tokens per expert to control it not out of the expert's capacity. Route tokens to experts using torch.einsum
- **Transformer Layers in TransformerBlock()**: The complete Transformer block is constructed to include: Multi-head self-Attention Layer, Switch MoE Layer, Residual Connections, Layer Normalization and Dropout Regularization.

1.3 Training with FGM(Fast Gradient Method)

- **FGM (Fast Gradient Method)**: It's employed to enhance the model's robustness against adversarial attacks: Compute the loss and backpropagate. Perturb the gradients of the embedding layer. Forward propagate again and accumulate gradients. Restore the original embedding parameters.
- **Loss function in training**: The training loop optimizes a composite loss function consisting of Cross-entropy loss for classification and auxiliary loss for the router's load balancing.

$$L = L_{\text{cross entropy}} + \lambda \cdot L_{\text{aux}}$$

- **Accuracy Optimization Techniques**: Introduce weight decay AdamW, label smoothing, early stopping and a learning rate scheduler in training progresses.

2 Experiment Details Result Analysis

Parameter Setting: A learning rate scheduler (ReduceLROnPlateau) is applied, reducing the learning rate from initial 0.005 by a factor of 0.5 if validation loss does not improve for 4 consecutive epochs. The parameters in Figure 1 reach the best result at 85.12%. Then conduct a comprehensive ablation study to investigate the impact of various training techniques on model performance in Figure 2.

Experiment 1: Learning Rate: Three different learning rates (0.001, 0.005, 0.01) were tested. The 0.001 shows slow steady improvement, but it failed to reach the peak performance of the 0.005 setting.

Parameter	Value
embed_dim	32
num_heads	2
ff_dim	32
num_experts	10
batch_size	50
learning_rate	0.005
dropout_rate	0.25
num_epochs	15
vocab_size	20000
num_tokens_per_example	200

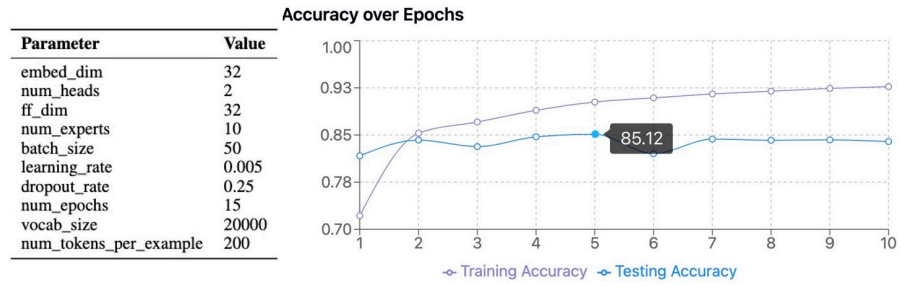

Figure 1: Experiment parameters setting and the best test accuracy result

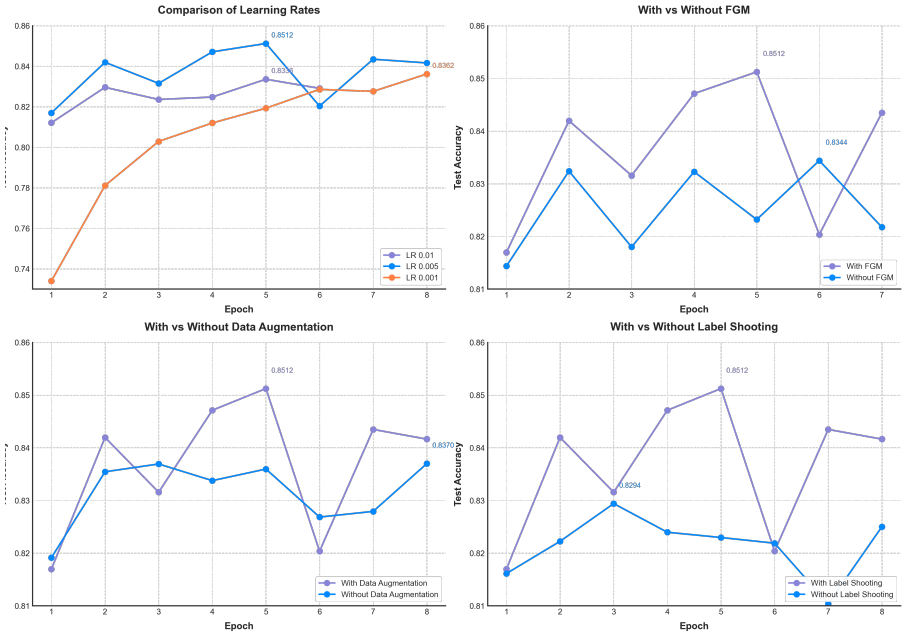

Figure 2: The ablation study to investigate the impact of various training techniques

The 0.005 shows rapid initial improvement, achieving the highest peak accuracy (0.8512) at epoch 5. The 0.01 shows quick convergence, unstable after epoch 5, training terminated after epoch 6.

Experiment 2: Fast Gradient Method (FGM): Evaluate the effect of FGM during training process. FGM consistently improved model performance, and the peak accuracy with FGM (0.8512) exceeded the peak without FGM (0.8344) by 1.68%, suggesting it enhance the model's robustness.

Experiment 3: Data Augmentation: Evaluate the effect of incorporating data augmentation techniques on generalization ability. It shows modest but consistent improvements in 5 out of 8 epochs. The augmented model also exhibited less volatility in accuracy across epochs, suggesting improved stability.

Experiment 4: Label Shooting: Evaluate the effect of label shooting which is a regularization technique. Label shoooting the most substantial impact among all tested techniques. The peak accuracy improved by 2.18%. Furthermore, the non-label-shooting configuration showed significant accuracy degradation after epoch 3, indicating poorer generalization.

Concolusion: The learning rate of 0.005 demonstrated the best overall performance. Label shooting provided the most substantial performance gain, followed by FGM, which highlight the importance of regularization techniques in preventing overfitting and enhancing generalization capabilities.

Future work: Explore if RL method in Deepseek-R1 like GRPO(Group relative policy optimization) will further optimize model performance.