RL Project Report

Siddharth Kothari (IMT2021019)

Sankalp Kothari (IMT2021028)

M Srinivasan (IMT2021058)

May 11, 2025

Contents

1 Problem Statement

Through this project, we explore the integration of sparse, structured masking (inspired by the PackNet approach) into a Mixture of Experts (MoE) framework. The goal is to enable multiple task-specific expert sub-networks to co-exist within a shared architecture without destructive interference, especially in continual learning or multi-task learning setups. The underlying idea is to use masks learned via parameterized matrices (using LU decomposition) to enforce sparse, expert-specific activations within shared layers. The architecture builds on a pre-trained ResNet-18 feature extractor, with expert-specific masked MLP layers and a shared classifier. We are currently attempting to get this model working on a set of tasks, each having their own set of classes.

2 PackNet and Adaptive Mixture of Local Experts (AMoLE)

2.1 PackNet: Iterative Pruning for Lifelong Learning

PackNet [1] addresses the challenge of *catastrophic forgetting* in sequential or continual learning. It proposes a method to incrementally add multiple tasks to a single neural network without requiring data from previous tasks. The approach relies on the observation that deep networks are typically overparameterized and many weights can be pruned without sacrificing performance.

The method consists of:

1. Training the model on the first task.
2. Applying magnitude-based pruning to identify and remove a certain percentage of unimportant weights.
3. Freezing the remaining important weights for the task.
4. Reusing the freed weights to learn new tasks.

This process is repeated for subsequent tasks, with a binary mask stored for each task to identify the active parameters.

Strengths:

- Efficient reuse of parameters across multiple tasks.
- Does not require retraining on old task data.
- Prevents interference between tasks by isolating their parameter subsets.

Weaknesses:

- Limited scalability — model capacity restricts the number of tasks.
- Does not promote shared learning between tasks.
- Requires mask storage and careful task scheduling.

2.2 Adaptive Mixture of Local Experts (AMoLE)

The **Adaptive Mixture of Local Experts** (AMoLE) model [2] is an early and influential architecture for divide-and-conquer learning. It proposes decomposing complex problems into simpler sub-problems, each handled by a dedicated "expert" network. A separate "gating" network learns to softly assign inputs to these expert networks.

The architecture is composed of:

- **Expert Networks:** Specialized neural networks, each focusing on a different region of the input space.
- **Gating Network:** A softmax-based network that outputs a probability distribution over the experts based on the input.

- **Mixture Output:** The final output is a weighted sum of all expert outputs, where weights come from the gating network.

The entire system is trained using gradient descent, with both the gating and expert networks learning simultaneously. The model uses a negative log-likelihood loss with a soft partitioning of data.

Strengths:

- Encourages specialization of experts and smooth interpolation between them.
- Learns both which expert to use and how to combine them adaptively.
- Improves generalization by focusing experts on subspaces.

Weaknesses:

- Training can be unstable if the gating network collapses (i.e., favors only one expert).
- Soft assignment can dilute specialization if not well-regularized.
- Experts do not evolve dynamically — structure is fixed after training.

2.3 Comparison and Insights

- **PackNet** partitions the parameter space statically after training each task, using pruning and freezing to isolate task-specific weights.
- **AMoLE** partitions the input space dynamically, using a gating network to assign inputs to a soft ensemble of expert models.
- **PackNet** is designed for *lifelong learning* with minimal interference between tasks; AMoLE is designed for *modular learning* to improve generalization within a single-task setting.
- **AMoLE** supports *simultaneous learning* across regions of the input space; PackNet focuses on *sequential learning* across tasks.

3 Motivation

The task of training a unified model across multiple diverse datasets presents key challenges, particularly in terms of preserving performance across domains and enabling task-specific specialization without interference. Our proposed approach draws inspiration from *PackNet* [1] and the *Adaptive Mixture of Local Experts (AMoLE)* framework [2].

From PackNet, we adopt the central idea of using task-specific binary masks to isolate subsets of the network weights for different tasks. In PackNet, after learning a task, weights critical to that task are retained via pruning, and subsequent tasks are learned using the unutilized parameters. This provides a static, hard partitioning of the parameter space to prevent catastrophic forgetting. However, this approach assumes task identity is known at inference time and lacks flexibility to dynamically route inputs to specialized sub-networks.

From AMoLE, we inherit the concept of learning to softly assign inputs to specialized expert networks using a *gating network*. AMoLE's strength lies in its ability to decompose a task into simpler regions of the input space and train local experts that specialize in those regions. The gating network allows for dynamic, input-dependent computation, enabling specialization without prior knowledge of the task label.

Combining the Strengths of Both Paradigms. Motivated by these two ideas, we propose a hybrid architecture that leverages:

- **Dynamic input routing** via a learnable gating network (as in AMoLE).
- **Expert-specific mask learning** that isolates parameters for different domains or datasets (as in PackNet).

Our insight is that in multi-domain or multi-dataset settings, each expert should learn to specialize on a different data distribution. Instead of predefining which expert should handle which dataset, we allow a learned gating network to perform this decision based on the input. Each expert network then learns its own binary mask over a shared parameter backbone. These masks control which weights the expert can update during training, allowing specialization while sharing the underlying architecture.

The architecture operates as follows:

1. During training, inputs from all datasets are provided without task labels.
2. A soft gating network predicts a distribution over experts for each input.
3. Each expert has its own mask that determines which parameters it can use and update.
4. The weighted sum of outputs from all experts (based on gating weights) is used for prediction.
5. The gating network, masks, and expert parameters are all jointly optimized.

Advantages of the Proposed Method:

- *Task-agnostic inference:* Unlike PackNet, our model does not require prior knowledge of which dataset or task the input belongs to.
- *Dynamic specialization:* Experts can specialize on different input regions or datasets as needed.
- *Parameter efficiency:* Experts share the backbone structure but isolate their computations via learned masks.
- *Continual scalability:* New experts can be added incrementally without retraining existing ones, by freezing their masks and parameters.

4 Model Architecture

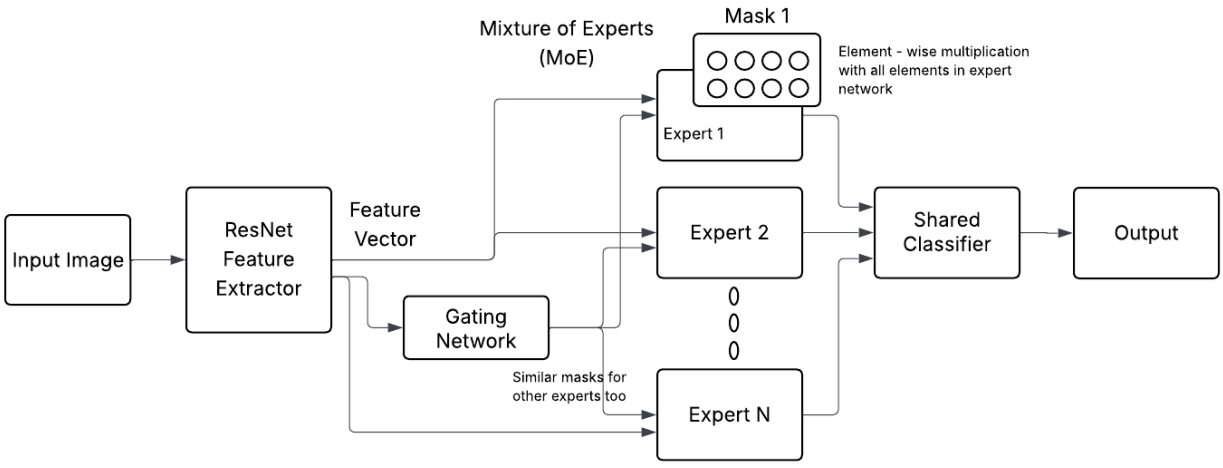

Figure 1: brief outline of the model

4.1 Feature Extractor

1. **Input**: Image tensor of shape $[3 \times 224 \times 224]$
2. **Output**: Feature vector of shape [512]

The FeatureExtractor class wraps a pre-trained ResNet-18 model with its final classification layer removed. It serves as a shared backbone across all experts, extracting high-level semantic features from the input image. In our Mixture of Experts (MoE) architecture, this shared feature extractor provides a common embedding space upon which the different expert modules will operate, enabling better comparability and gating decisions.

4.2 Gating Network

1. **Input**: 512-dimensional feature vector
2. **Output**: Probability distribution over num_experts (via Gumbel Softmax)

The GatingNetwork learns to route inputs dynamically to different experts based on the input features. It uses a single linear transformation followed by Gumbel Softmax to approximate categorical sampling in a differentiable manner. The Gumbel Softmax allows backpropagation to update both the expert modules and the gating mechanism during training. The output represents a soft assignment of each input to all experts, enabling a single expert to dominate. This probabilistic routing is crucial for specialization and diversity in the expert modules.

4.3 Masked LU Feature Extractor

1. **Input**: 512-dimensional feature vector
2. **Output**: 64-dimensional transformed representation per expert

The MaskedLUFeatureExtractor is the core component of our MoE experts. It comprises a 4-layer MLP, where each linear layer is sparsely masked. The LU-based masking approach aims to learn a low-rank approximation of task-specific sparsity, drawing inspiration from the PackNet method. Though potentially more powerful and interpretable, this method can be unstable during training. Each expert has independently initialized and masked parameters, allowing them to learn distinct mappings from the shared input features.

4.4 Shared Classifier

1. **Input**: 64-dimensional expert output
2. **Output**: Class logits for the different tasks

The SharedClassifier is a simple fully connected layer used by all experts to map their output representations into the final prediction space. Sharing the classifier ensures a consistent target representation across experts and supports ensemble behavior when multiple experts contribute to a prediction. This also simplifies evaluation and loss computation, as all outputs lie in the same class space. It promotes cooperation among experts during learning while still allowing them to specialize in feature extraction.

5 Experiments

We conducted a series of experiments using our MoE model. Below are the setup and results form each -

5.1 Two tasks, Two experts

Setup: We trained the model with two distinct tasks (datasets), using two dedicated experts and the shared feature extractor and classifier. The LU-based masked expert networks were expected to specialize per task via gating.

Observations:

Table 1: Training Loss on Combined CUB + Flowers Dataset
(2-data, 2-experts, No Mask Training)

Batch	Loss	Batch	Loss
0	5.2875	10	5.2978
20	5.2846	30	5.3006
40	5.2894	50	5.3063
60	5.2962	70	5.2967
80	5.3065	90	5.3122
100	5.3140	110	5.2971
120	5.2973	130	5.2984
140	5.2899	150	5.2978
160	5.2932	170	5.2993
180	5.3036	190	5.2949
200	5.2984	210	5.3022
220	5.3049		

1. The overall classification accuracy was poor across both tasks.
2. The model exhibited circling loss behavior—a sign of instability where training oscillates without convergence.

Inference and Analysis: We believed the possible reasons for these observations to be -

1. The LU-based masks in the expert network may have introduced rigidity or optimization difficulty.
2. Gating network may not be able to learn how to route the image from one task to one expert, maybe due to gradient not propagating back. *To check this we tried one task but with 2 experts. The idea is that we should get no images being routed to one of the experts.*

5.2 One task, Two experts

Setup: We simplified the setup to a single task but retained two experts, aiming to verify the correctness of the gating network.
Observations:

1. The gating network correctly converged to using only one expert, routing all inputs to the same path. This behavior matches expectations in a single-task setup, where specialization is unnecessary.
2. However, despite correct routing, the final classification performance remained poor.

Implications: The issue likely lies deeper—potentially within the expert module itself (i.e., masking strategy or training flow) or the classifier's adaptation to masked representations.

5.3 One Task, One expert

Setup: We used a single task and a single expert with the full LU-based learnable masking approach active.
Observations:

1. Model accuracy still remained poor.
2. Despite removing the gating network from the equation, the LU-masked expert failed to learn adequate representations.

Table 2: Training Loss on CUB
(1-expert-1-task, No Mask Training)

Batch	Loss	Batch	Loss
0	5.3017	100	5.3006
10	5.2892	110	5.2990
20	5.2972	120	5.2827
30	5.2939	130	5.3083
40	5.3006	140	5.2859
50	5.3076	150	5.3103
60	5.3025	160	5.2876
70	5.3066	170	5.2865
80	5.2974	180	5.3028
90	5.2965		

Table 3: Training Loss on Combined CUB + Flowers Dataset
(2-expert-2-task)

Batch	Loss	Batch	Loss
0	5.3012	10	5.2990
20	5.2967	30	5.2984
40	5.2992	50	5.3040
60	5.3123	70	5.2965
80	5.2966	90	5.2962
100	5.2926	110	5.3059
120	5.3020	130	5.2970

Possible Reasons:

1. Poor initialization, gradient flow obstruction due to hard masking, or ineffective learning of the LU components might be responsible.

2. To check whether our model itself seems to be correct, we ran one final experiment - removing the learnable LU mask. This would help us check if there is any problem withe the model itself.

5.4 One task, One expert with hard-coded mask

Setup: In this controlled experiment, we completely bypassed the LU-based mask and instead used a fixed binary mask with 50% sparsity, hard-coded into each layer of the expert.
Observations: No improvement in performance was observed compared to earlier setups. The same issue still persists.
Analysis:

1. This outcome suggests that the problem is likely not solely in the learnable masking logic.

2. It raises questions about the interaction between masked representations and the classifier during joint training.

Table 4: Training Loss on CUB
(1-task-2-experts)

Batch	Loss	Batch	Loss
0	5.3017	100	5.3006
10	5.2892	110	5.2990
20	5.2972	120	5.2827
30	5.2939	130	5.3083
40	5.3006	140	5.2859
50	5.3076	150	5.3103
60	5.3025	160	5.2876
70	5.3066	170	5.2865
80	5.2974	180	5.3028

6 Removing Gating and Masking (Pure ResNet and Expert layer without masking) with Cifar-10

Setup: In this diagnostic experiment, we removed both the gating network and the masking mechanism. Only the base ResNet model and 4 layer network was used for forward propagation, without any expert selection or sparsity control. This was done to isolate and determine whether the issues observed in prior configurations were stemming from architectural components or a deeper implementation error. It exhibited good performance, which suggested that the way we were reading the data previously was wrong, which we fix later. The results for cifar-10 are shown in table 5. We ran it for 8 epochs, after which performance was saturated.

Epoch	Loss	Accuracy (%)
1	0.8284	71.25
2	0.4572	85.13
3	0.3281	89.44
4	0.2554	91.76
5	0.2027	93.41
6	0.1582	94.84
7	0.1295	95.85
8	0.1095	96.58

Table 5: Epoch-wise Loss and Accuracy on Cifar Dataset

7 Cubs Performance after Removal of Gating Network and Masking

From the previous experiment, we realised **there were issues with the way we were reading the dataset**, as the inbuilt pytorch dataset worked pretty well. We debugged and fixed those errors, and then we ran the same model without the gating network and masking to confirm that now the cubs dataset is being read correctly. We noticed that the accuracy and loss constantly increasing and decreasing respectively. This motivated us further to first run our experiments with fewer layers of expert network (2 layers) and then see the output. The output seemed really good of achieving around 91% accuracy for 20 epochs, the output per epoch is shown in Table 6.

Epoch	Loss	Accuracy (%)
1	4.7830	2.74
2	3.8224	8.61
3	3.2905	14.53
4	2.8247	22.34
5	2.4026	31.48
6	2.0186	39.66
7	1.6760	47.31
8	1.3239	57.24
9	1.1245	63.95
10	0.9174	69.34
11	0.7254	75.46
12	0.6057	79.76
13	0.5318	82.17
14	0.4843	84.12
15	0.3822	87.45
16	0.3300	89.61
17	0.3345	88.87
18	0.2482	92.04
19	0.2734	91.29
20	0.2459	92.14

Table 6: Epoch-wise Loss and Accuracy on 2 layer Network without masking and gating

We also ran the model with the previously decided 4 layers, and the results are shown in Table 7.
Observations: Now we can see that the model is slowly improving and learning the weights for that particular dataset. We reached an accuracy of about 37% with 20 epochs and 4 layers. We feel that increasing the epochs will further increase the accuracy of the model and lower the loss, as from the table we observe a steady increase from 2 % to 37 % accuracy, which we feel would be more given more epochs.

The results after running on more epochs (50) is shown in Figure 4. Note that this was a separate experiment, and hence the values will definitely slightly vary for per epoch loss and accuracy.

8 Experiments Rerun

8.1 One task, One expert with hard-coded mask with 50 % sparsity

Setup: In this controlled experiment, we completely bypassed the LU-based mask and instead used a fixed binary mask with 50% sparsity, hard-coded into each layer of the expert. This mask is non learnable.

The results for the experiment are shown in Figure 3. The similarity in performance to the no mask case suggests that the model is able to recover from the sparsity pretty well, and is able to achieve similar performance. This was expected, as even with the 2 layers, the model had similar performance, suggesting that it requires much less parameters than what are being given to it, in order to learn effectively.

8.2 One Task, One expert with learnable LU-Mask

Setup: We used a single task and a single expert with the full LU-based learnable masking approach active.

Epoch	Loss	Accuracy (%)
1	5.1860	0.75
2	4.8802	1.20
3	4.6851	2.19
4	4.5262	2.85
5	4.4212	2.84
6	4.2861	3.74
7	4.1443	4.47
8	4.0185	5.51
9	3.8974	6.56
10	3.7770	7.84
11	3.6633	8.94
12	3.4683	10.94
13	3.2588	13.71
14	3.0888	15.80
15	2.9279	18.57
16	2.7260	21.96
17	2.5686	24.69
18	2.3930	28.80
19	2.1820	32.42
20	1.9925	37.64

Table 7: Epoch-wise Loss and Accuracy on 4 layer Network without masking and gating, run for 20 epochs

We observe slow but strong learning of the mask, as shown by the steady increase till 88 % accuracy in figure 3. This suggests the model is able to learn a mask alongside the actual parameters ad weights to learn.

8.3 Two Tasks, Two Experts with Learnable LU-Mask

Setup: We used two tasks and two experts, each associated with one task, and activated the LU-based learnable masking for both.

We observe a clear and consistent learning pattern over the epochs. From Fig 5, the accuracy steadily climbs from a mere 0.60% at epoch 1 to approximately 88% by epoch 48, before slightly stabilizing. This indicates strong convergence behavior and shows that the LU-based masking approach successfully supports disentangled learning for multiple experts in a multi-task setting. Additionally, the continuous decrease in loss demonstrates effective joint optimization of both expert parameters and their respective masks.

8.4 Key Finding

A key observation from our experiments was that the 1-data 1-expert model performed exceptionally well on the training data, achieving a low loss of 0.2557 and a high accuracy of 92.04%. However, its performance on the test data was significantly worse, with a loss of 8.3495 and an accuracy of only 17.19%. This stark contrast indicates severe overfitting, where the model has learned the training data too well but fails to generalize to unseen data. The same issue is also observed in 2 expert 2 task setting. Addressing this overfitting is our primary objective moving forward.

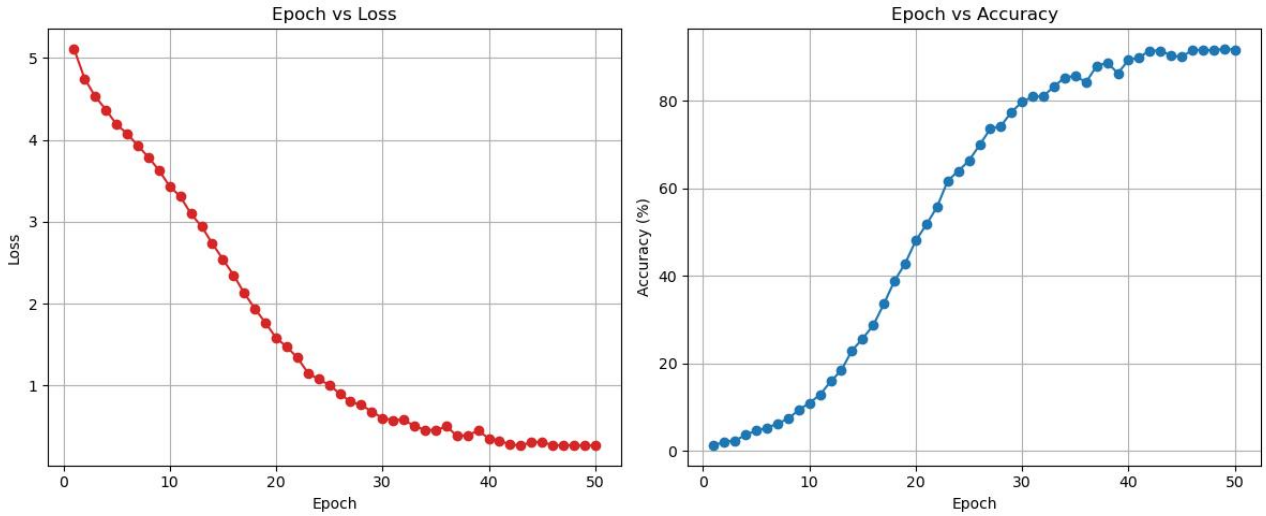

Figure 2: Training on cubs dataset with a 4 layer neural network for 50 epoch

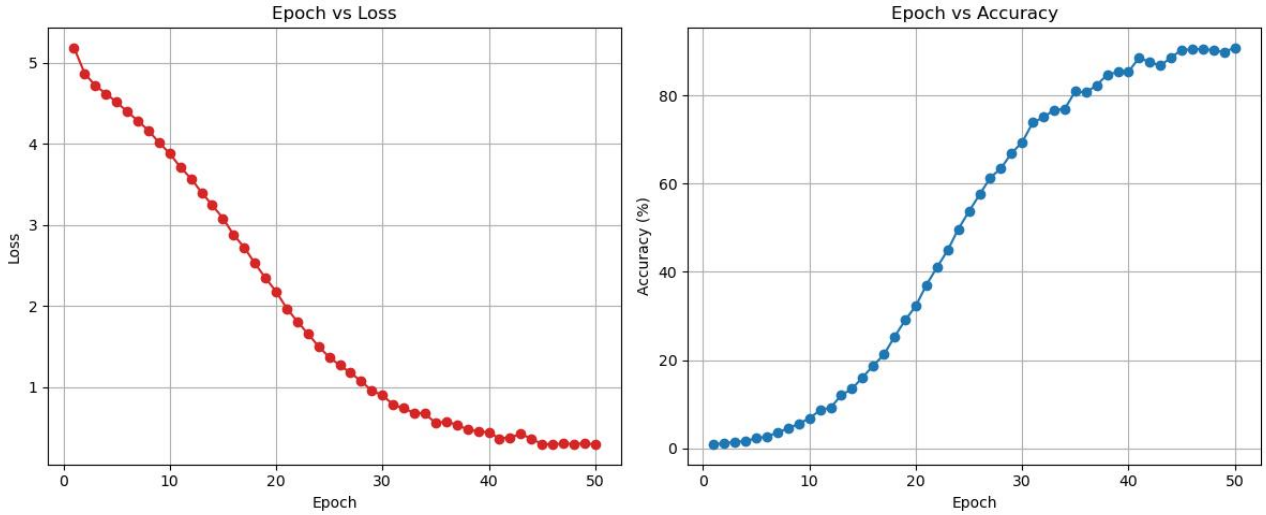

Figure 3: Training on cubs dataset with a mask with 50 % sparsity for 50 epochs

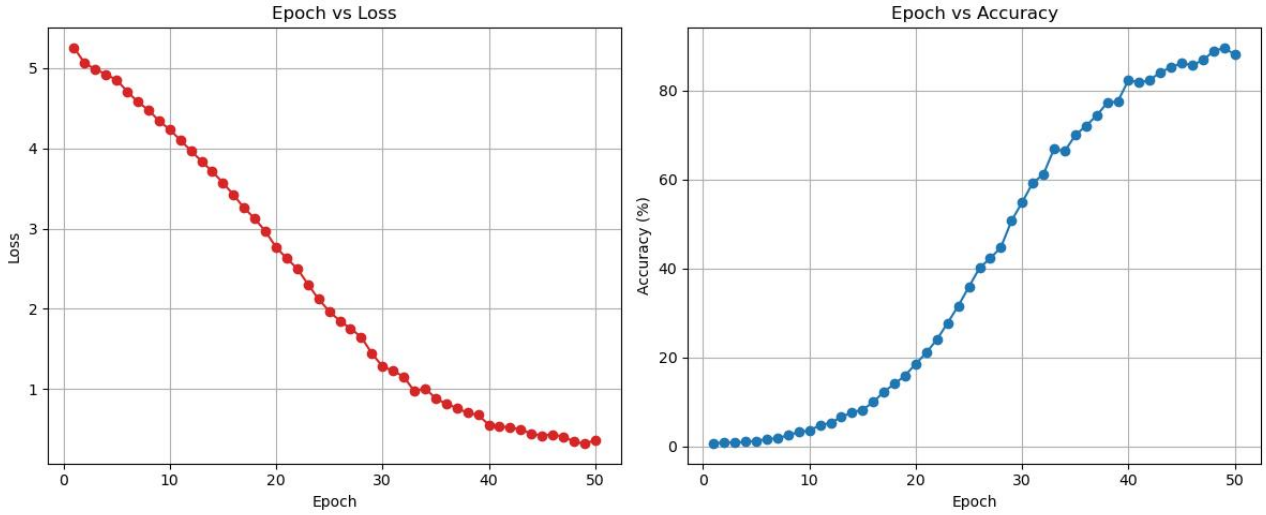

Figure 4: Training on cubs dataset with a **learnable mask using LU decomposition** for one expert and one task for 50 epochs

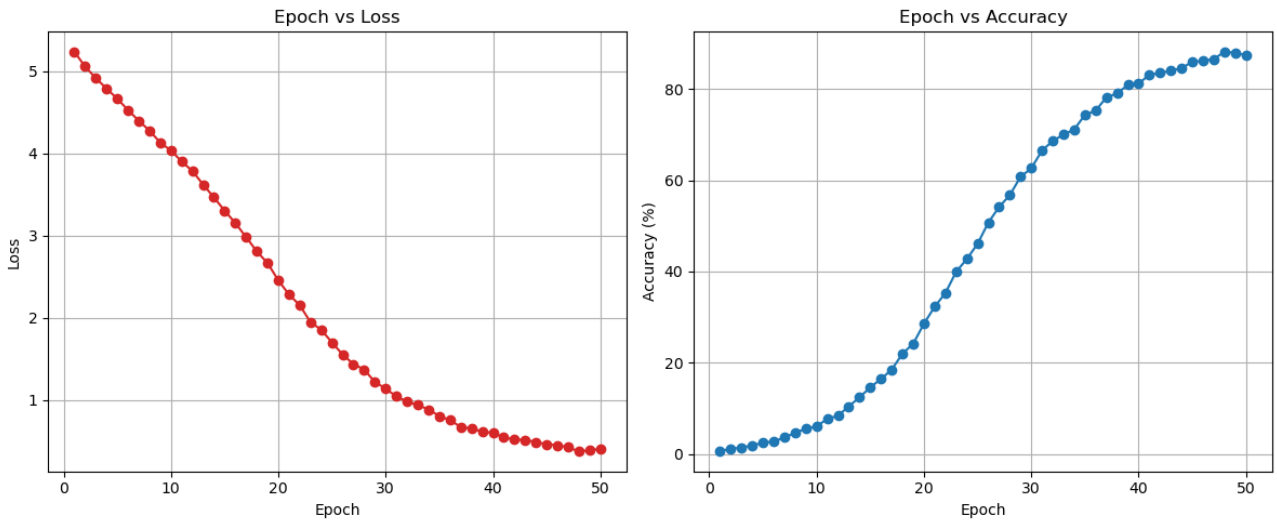

Figure 5: Training on cubs dataset and flower dataset with a learnable mask with for 50 epochs

9 Future Work

As of now, we now have a working code with correct dataset parsing and a working code (with one expert and one task at least). We now wish to do the following in the future:

1. **Rectify Overfitting:** Add metrics to ensure that the model stops early and prevents overfitting.
2. **Rerun the other experiments with the correct code:** We wish to run the previous experiments as well, whose results are currently outdated, and check the quantitative performance in multiple settings.
3. **Qualitative Analysis and Visualisations:** Carry out an analysis of the gating behaviour and the masking behaviour for the setting with 2 tasks and 2 experts, and visualise the training results, and quanitfy the same.
4. **Ensure Proper Gating:** Check the routing of the images, and confirm whether all the images of one dataset type (example cubs or flowers) are routed to the same expert or not.
5. **Add more Datasets:** Add a third dataset and check the performance in that case, and check whether the performance remains the same or not, similar to the PackNet [1] paper.
6. **Experiment with Masking on ResNet:** Observe and analyze the performance when the masking is applied not only to the final layers, but to the resnet backbone as well, and compare outcomes.

References

[1] A. Mallya and S. Lazebnik, "Packnet: Adding multiple tasks to a single network by iterative pruning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7765–7773, 2018.

[2] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," in *Neural Computation*, vol. 3, pp. 79–87, MIT Press, 1991.