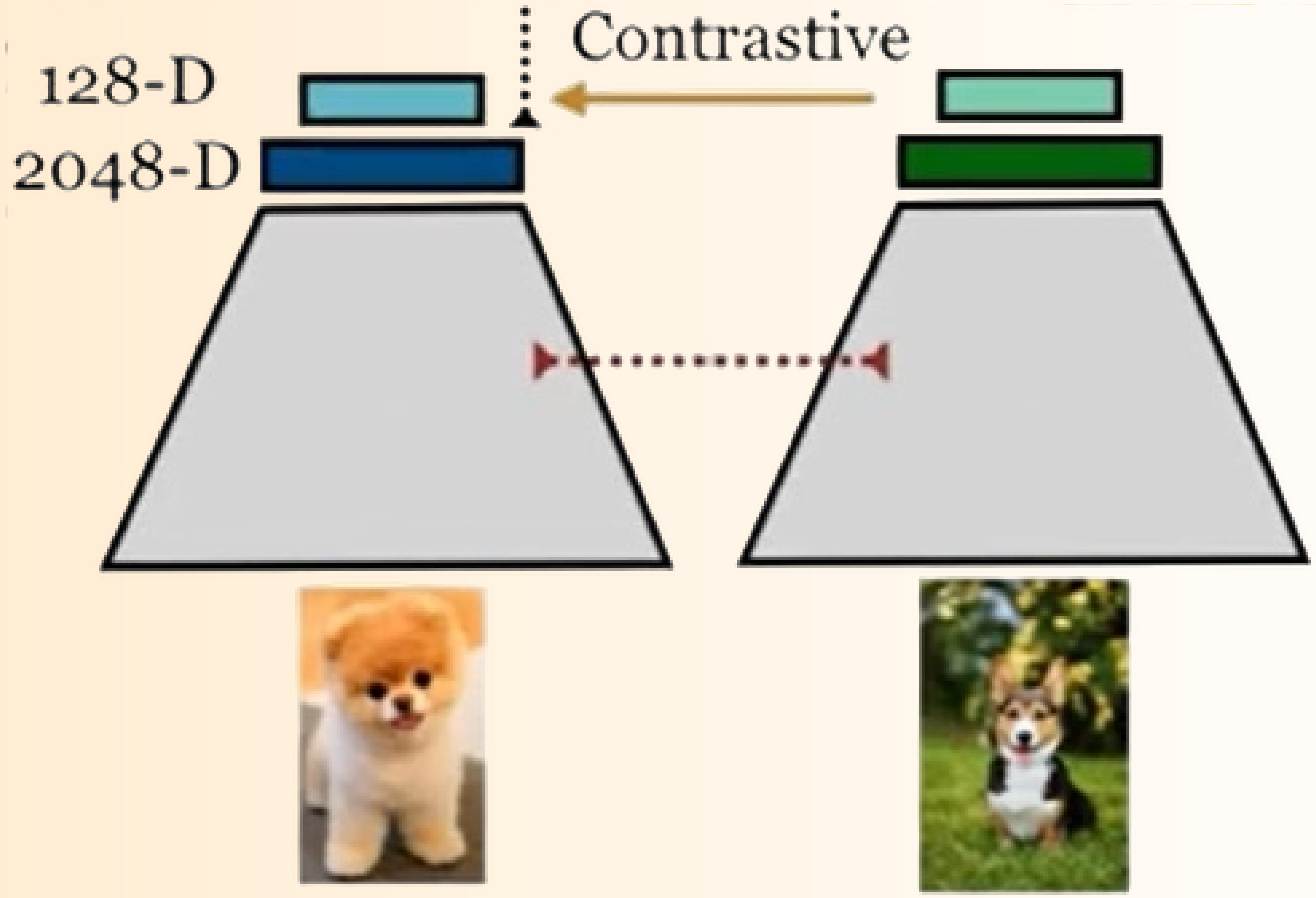
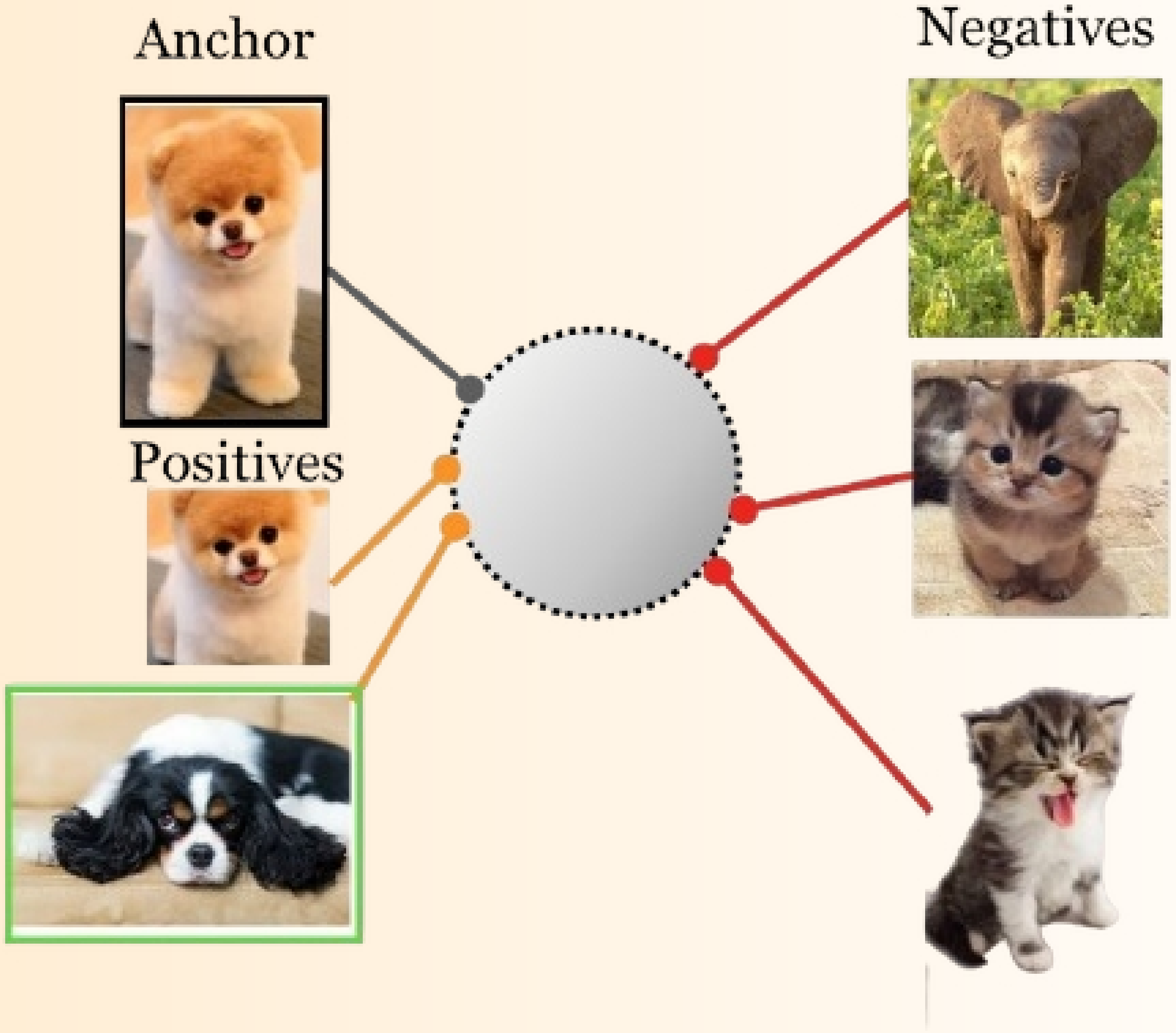
TRACK - 1 Supervised Contrastive Learning



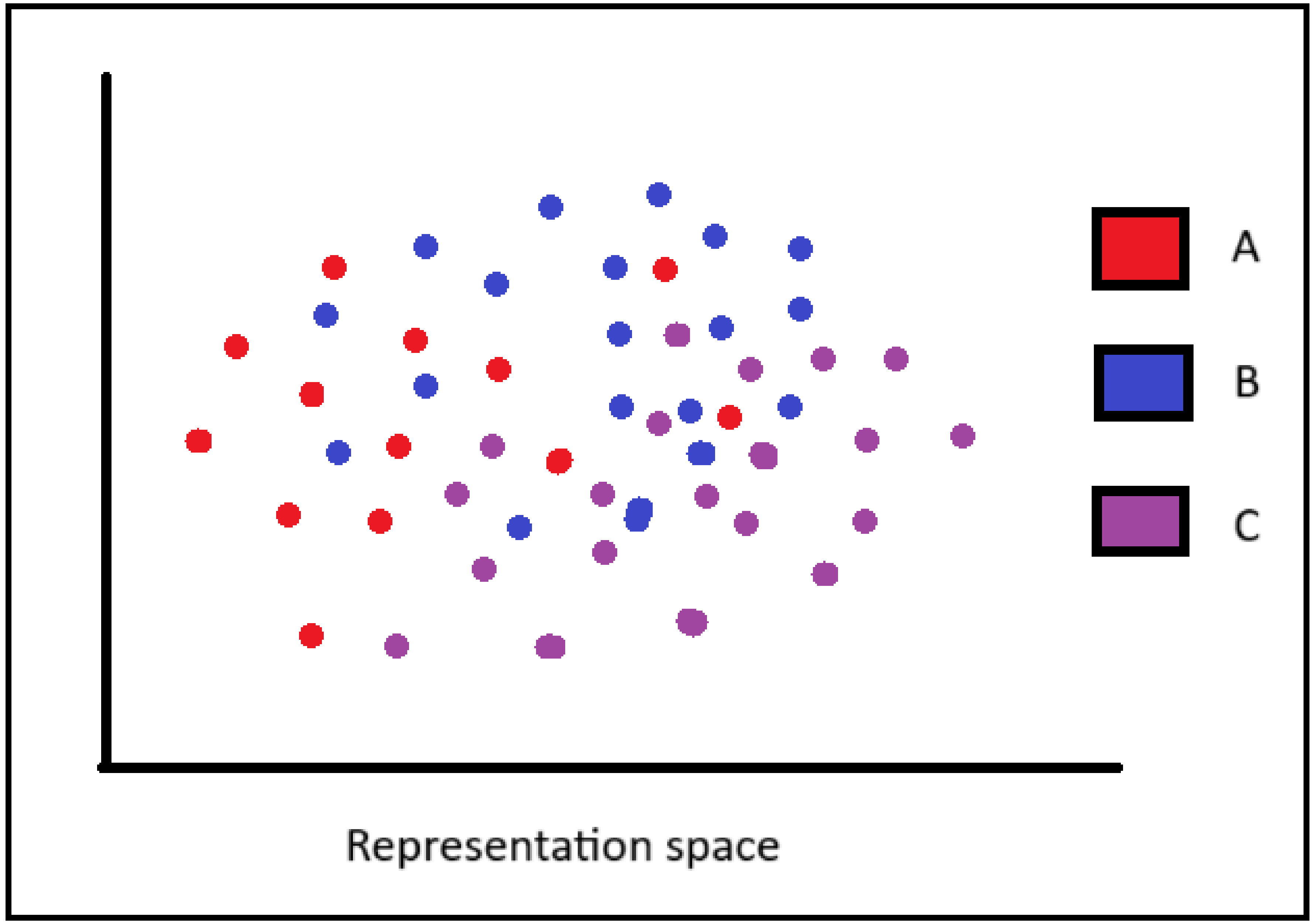
Why Supervised Contrastive Learning?



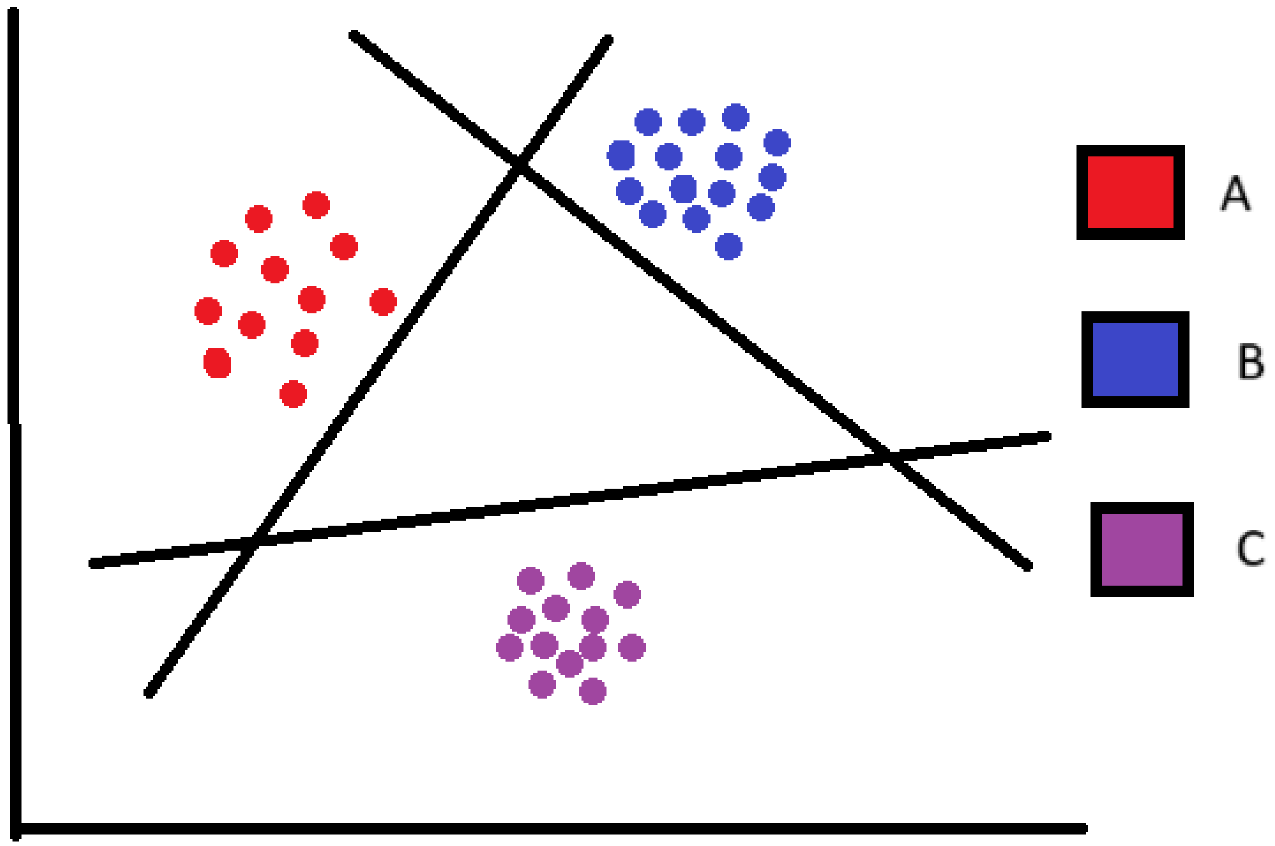
Supervised Contrastive

Problems Without Contrastive Training

- No clear demarcation between classes
- Difficult to be classified by linear classifiers



Representation space after contrastive pre-training

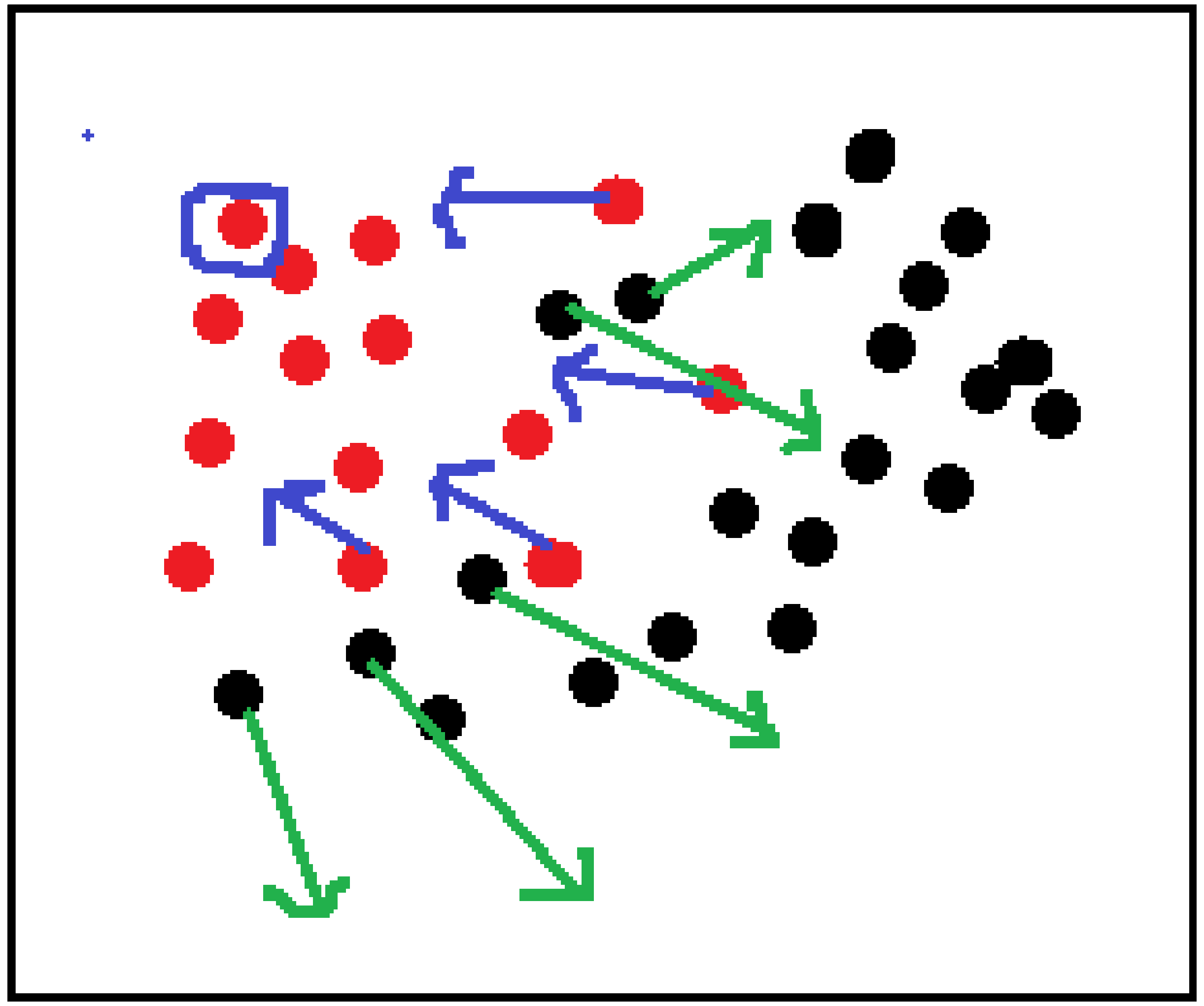


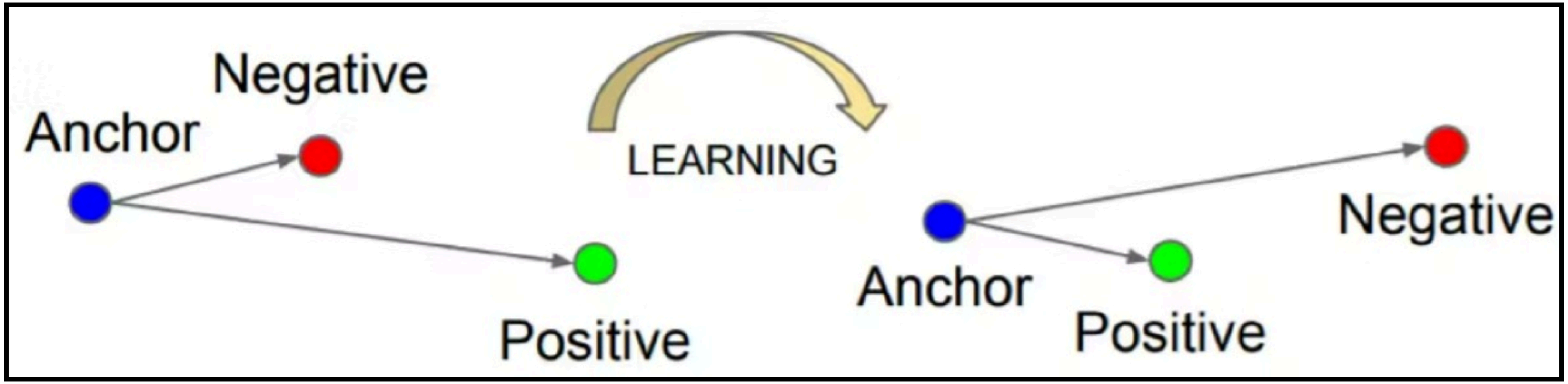
Representation space
(after contrastive learning)

- After contrastive separation, the data gets fragmented into clusters in the respresentation space.
- This being clustered and separated makes it easier for linear classifiers.
- Thus, it gives better accuracy.
- It also requires less number of epochs for training.

How does it do that?

- We pick an anchor and push all positive samples towards it.
- We simultaneously "push" all the negative samples away from our anchor in the representation space.
- This clusters similar data together.





Contrastive Loss Function:

For each anchor, the loss tries to:

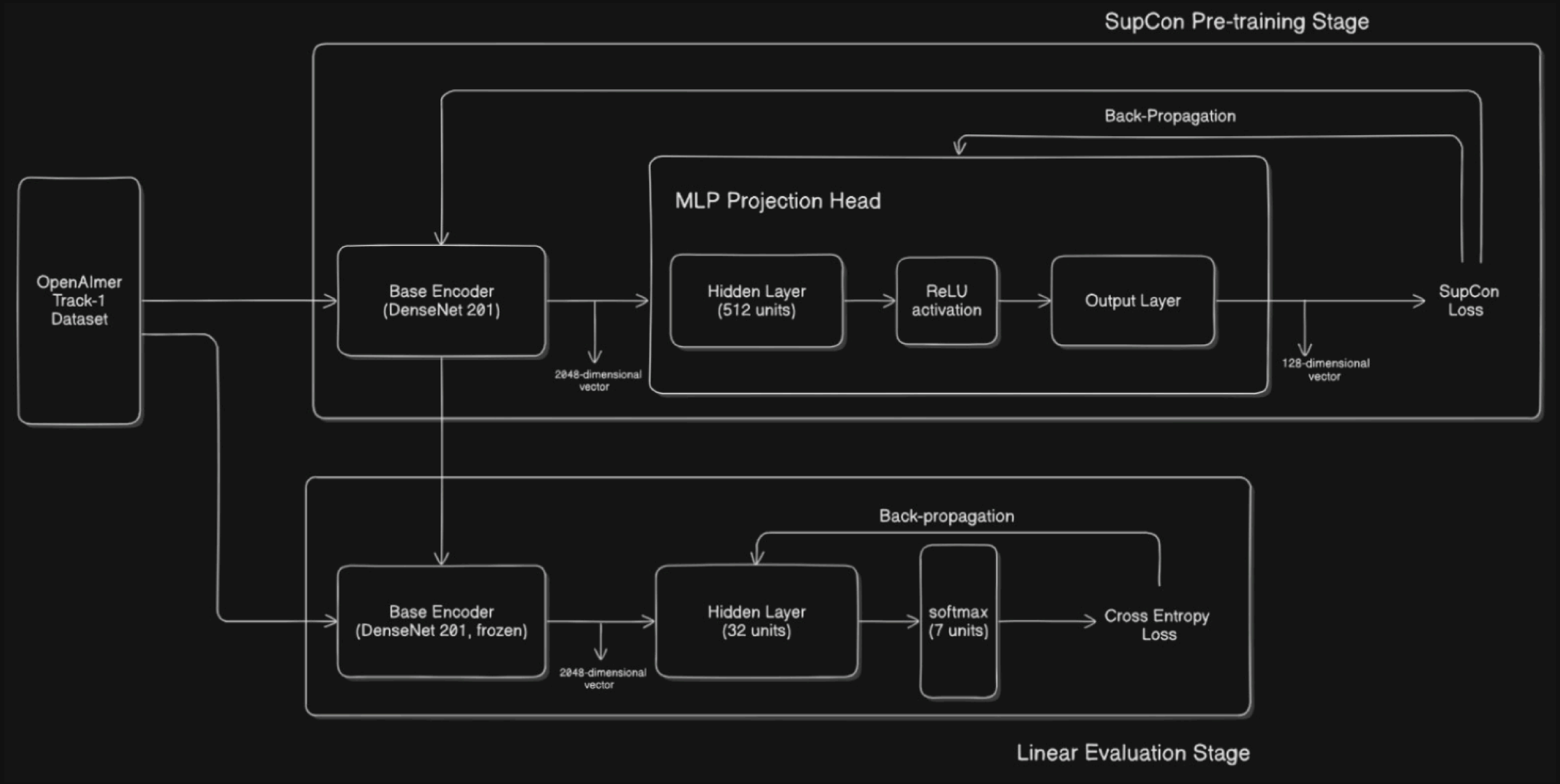
- Pull its embedding z_i closer to all of its positive samples Z_p (same class).
- Push z_i away from all other samples Z_a in the batch, regardless of class.

This is done via a log-softmax term, promoting high similarity for positive pairs and low similarity for others.

$$\boxed{\mathcal{L}_{in}^{sup} = \sum_{i \in I} \mathcal{L}_{in,i}^{sup} = \sum_{i \in I} -\log \left\{ \frac{1}{|P(i)|} \sum_{p \in P(i)} \frac{\exp(\boldsymbol{z}_i \bullet \boldsymbol{z}_p / \tau)}{\sum_{a \in A(i)} \exp(\boldsymbol{z}_i \bullet \boldsymbol{z}_a / \tau)} \right\}}$$

I : Set of indices for the anchor samples in the batch.
z_i : Normalized embedding of the anchor sample i.
P(i) : Set of indices of positives for anchor i (same class).
A(i) = I \ {i} : Set of all other samples except i (used for the denominator).
τ : Temperature parameter (scales similarity) — usually $\tau \in (0, 1]$.

PIPELINE



SupCon Pre-training Stage
Back-Propagation
MLP Projection Head
OpenAImer Track-1 Dataset
Base Encoder (DenseNet 201)
Hidden Layer (512 units)
ReLU activation
Output Layer
SupCon Loss
2048-dimensional vector
128-dimensional vector
Back-propagation
Base Encoder (DenseNet 201, frozen)
Hidden Layer (32 units)
softmax (7 units)
Cross Entropy Loss
2048-dimensional vector
Linear Evaluation Stage

Architecture Overview

- **Two-stage Framework:** i) SupCon Pre-training Stage ii) Linear Evaluation Stage.

- **Base Encoder:** DenseNet201 Architecture
  Used in both stages (trainable during SupCon stage, frozen during evaluation stage).

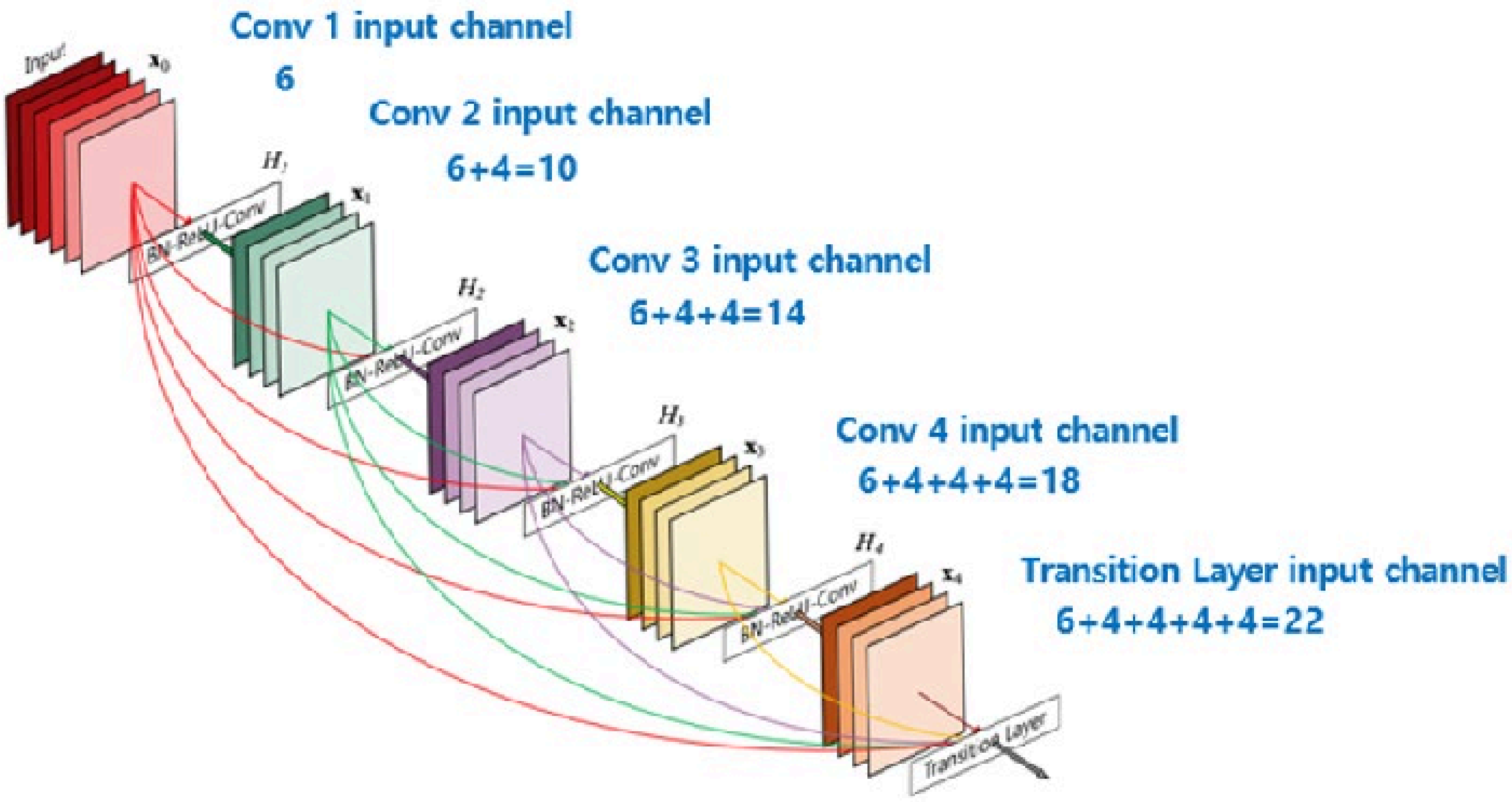
- **SupCon Pre-training stage:**
  The DenseNet201 encoder extracts features from the input images.
  Features passed through MLP Projection Head.
  Encoder trained using Supervised Contrastive Loss.

- **Linear Evaluation Stage:**
  Pre-trained encoder is frozen, its outputs are used as features
  Classifier (softmax activation) is trained using Cross Entropy Loss for final classification

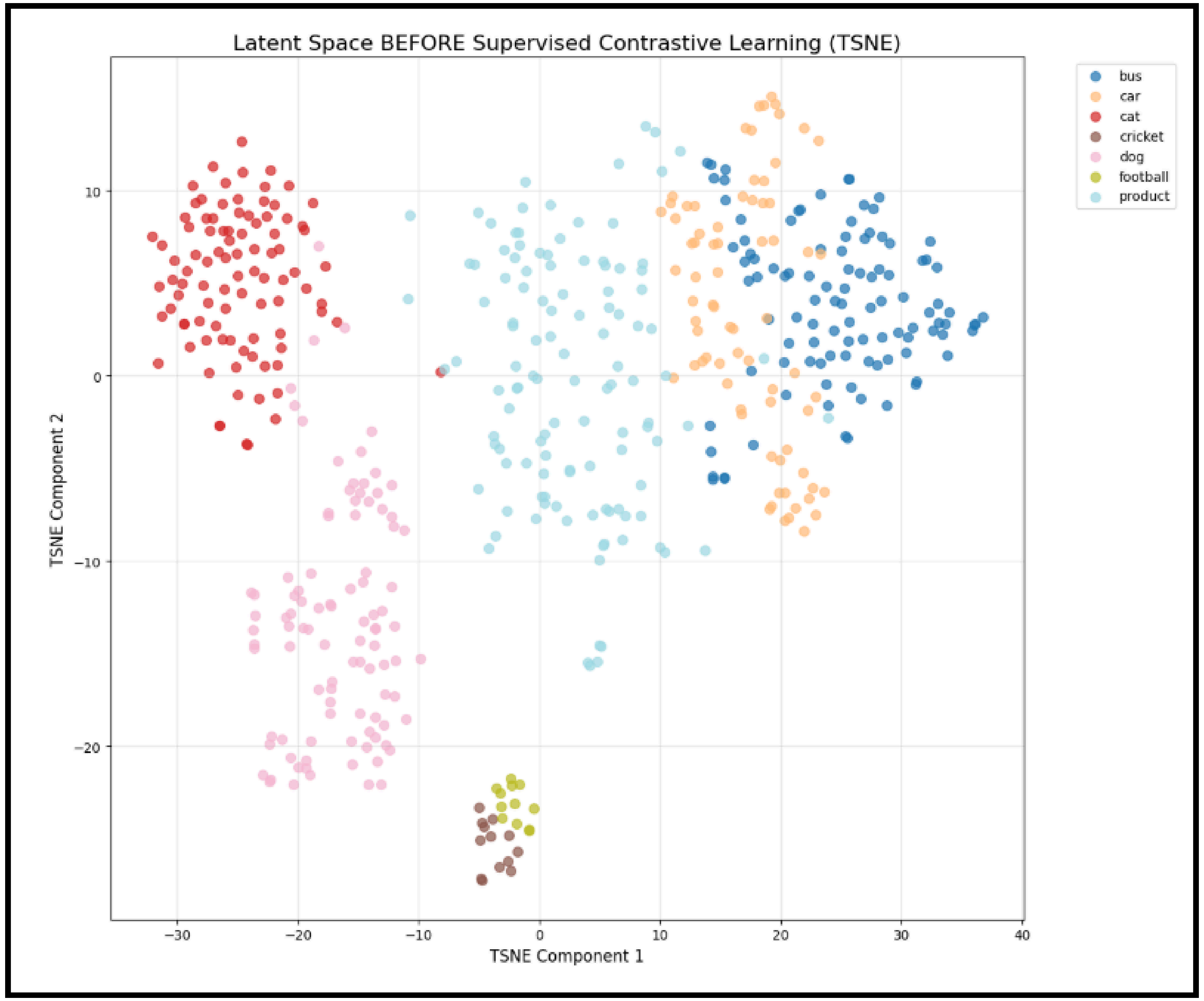
- **Back-propagation Strategy:**
  In pre-training: full back-propagation through encoder + projection head.
  In evaluation stage: back-propagation only through linear classifier.

DenseNet201 Model Statistics:

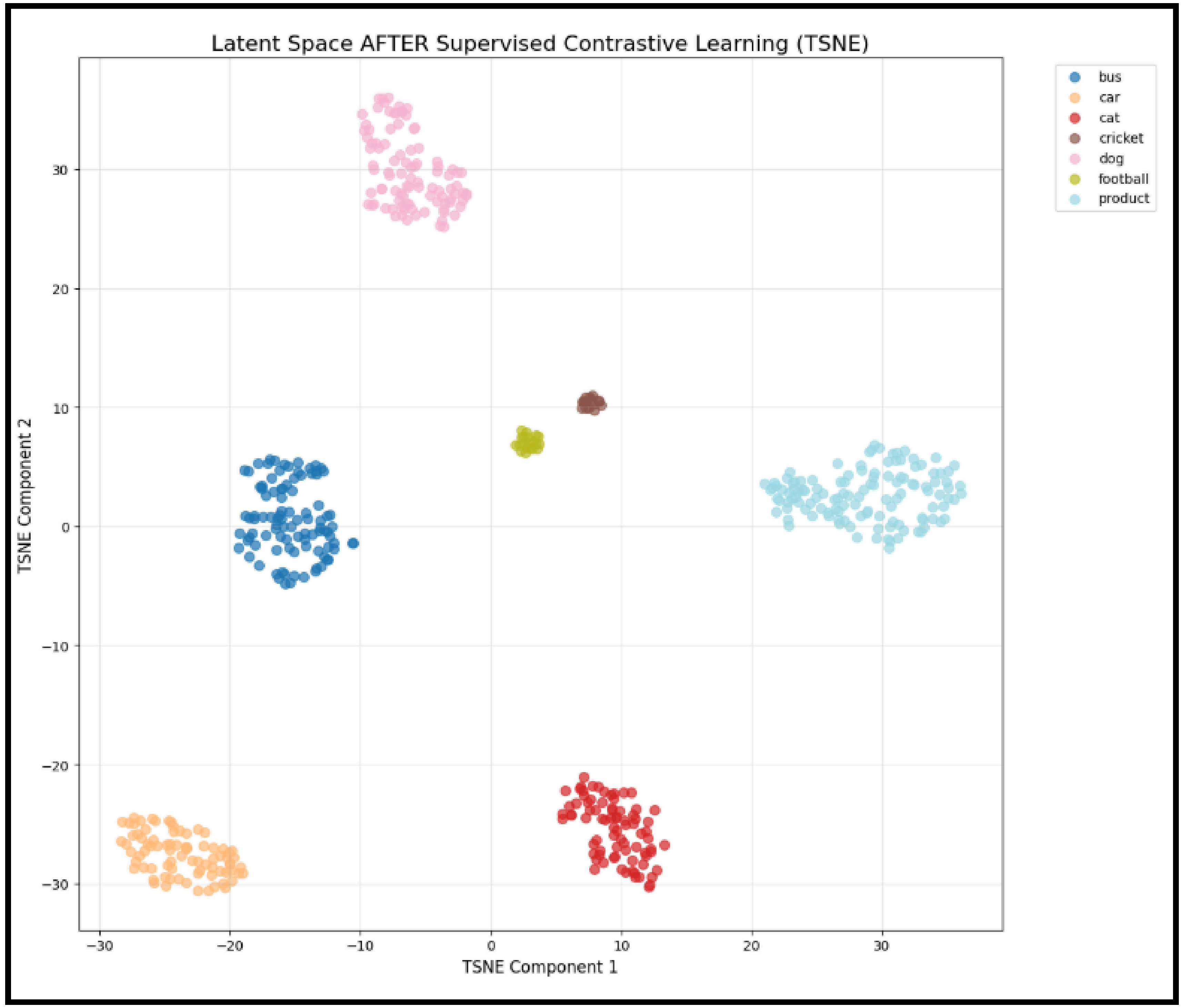
- F1-Score: 0.9948
- Accuracy: 0.9936
- Precision: 0.9944
- Recall: 0.9952



t-SNE Plots

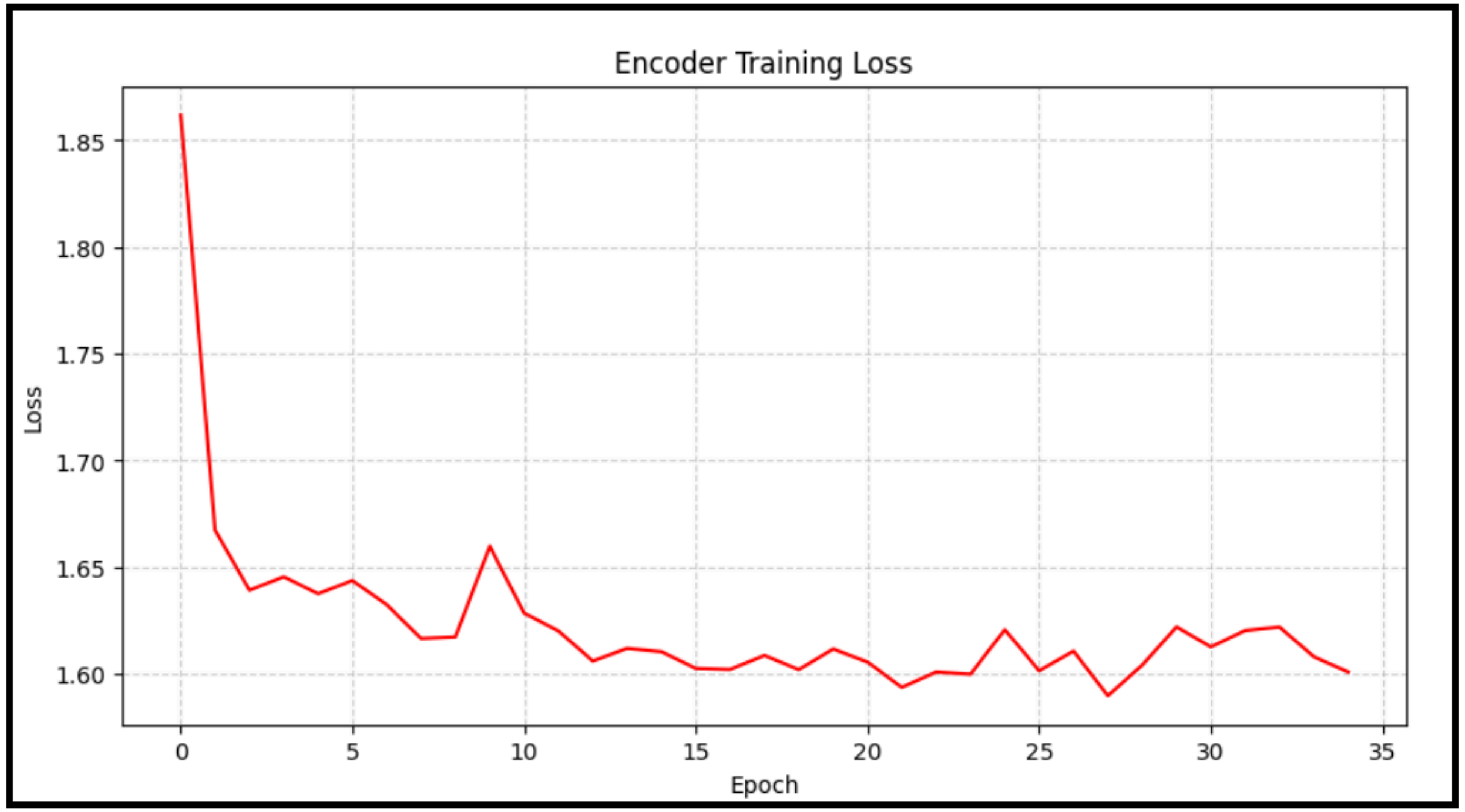


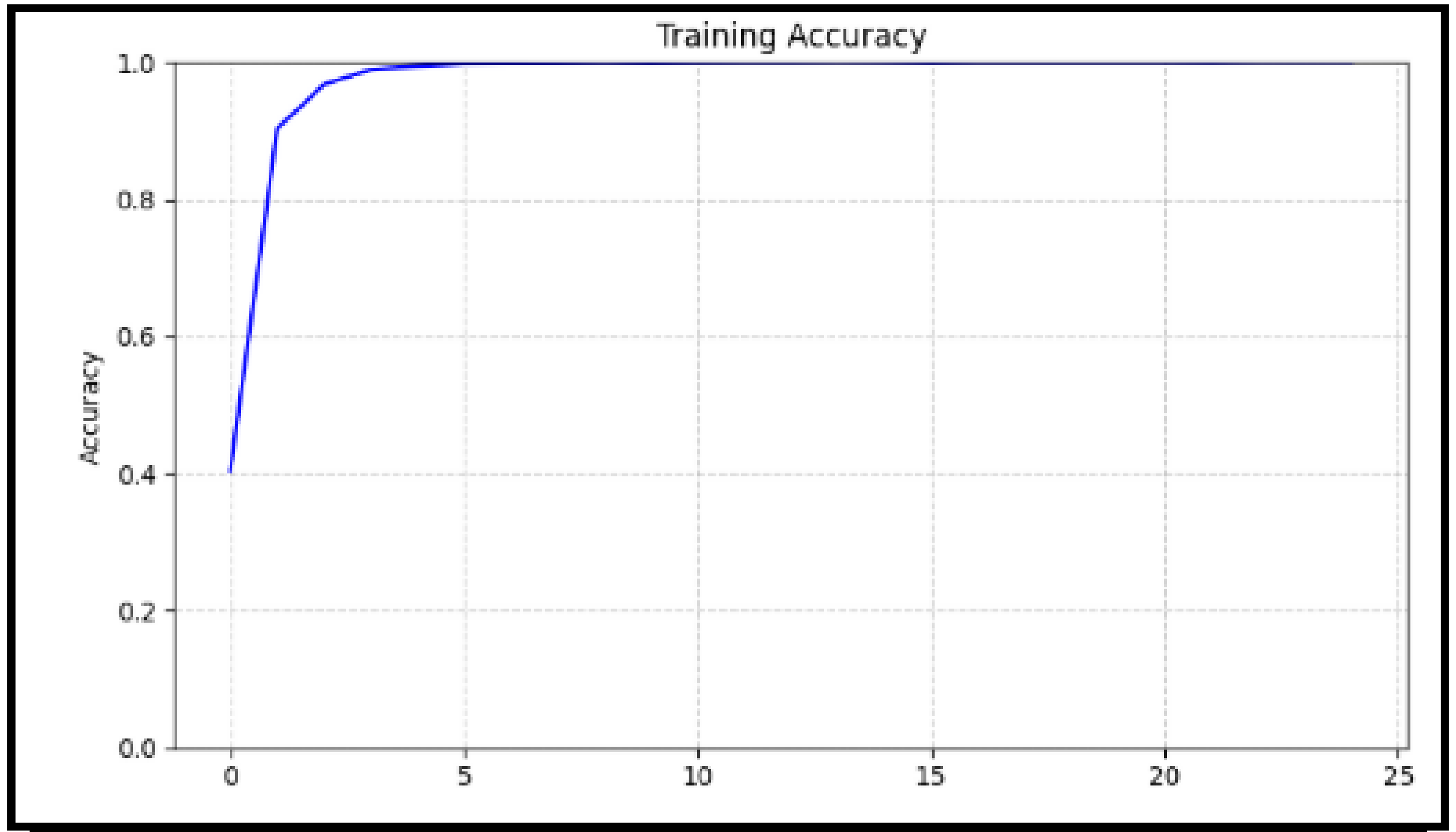
Latent Space BEFORE Supervised Contrastive Learning (TSNE)
bus
car
cat
cricket
dog
football
product
TSNE Component 1
TSNE Component 2
−30
−20
−10
0
10
20
30
40

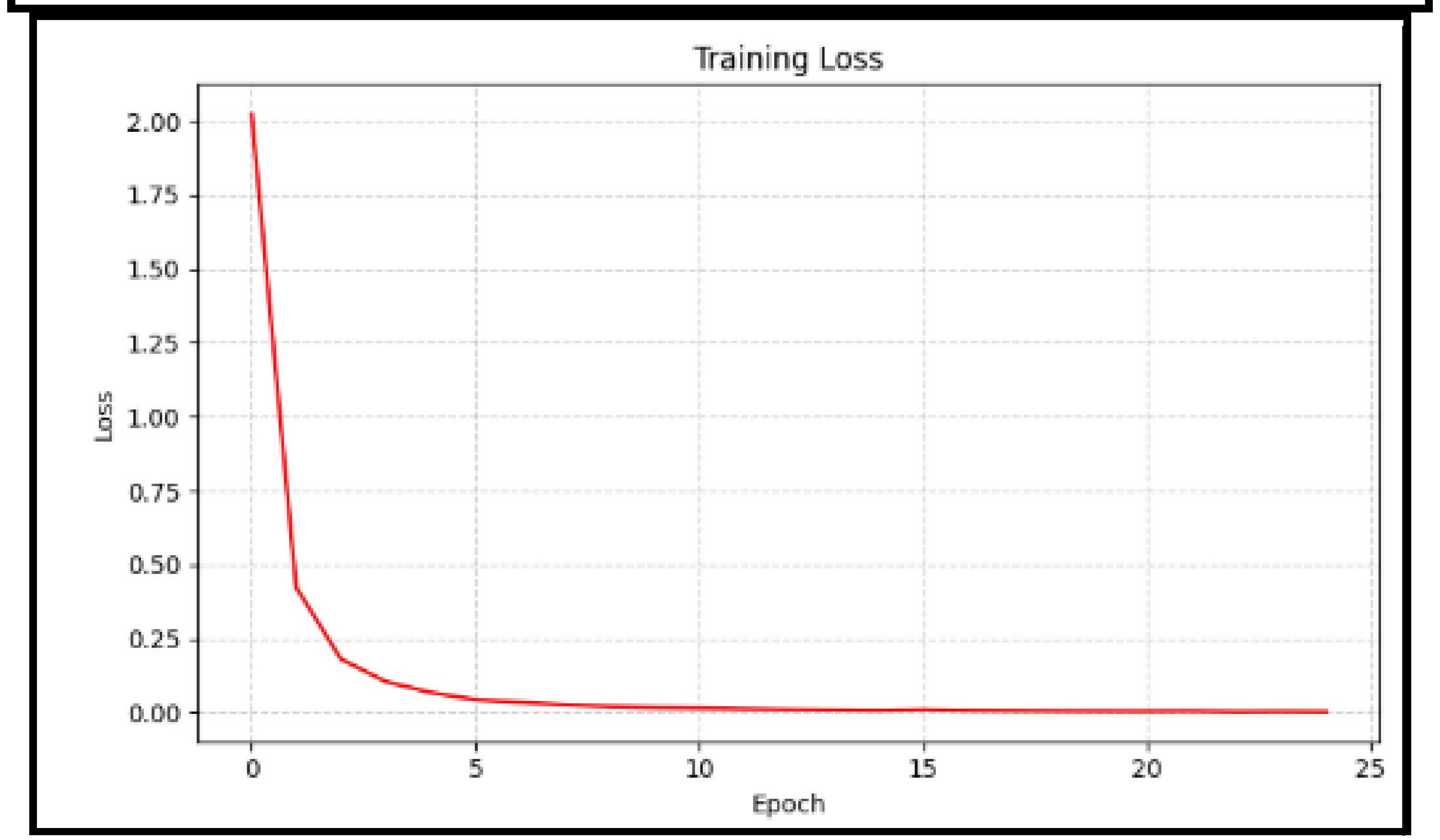


Latent Space AFTER Supervised Contrastive Learning (TSNE)
bus
car
cat
cricket
dog
football
product
TSNE Component 1
TSNE Component 2
−30
−20
−10
0
10
20
30

Encoder and Classifier Performance







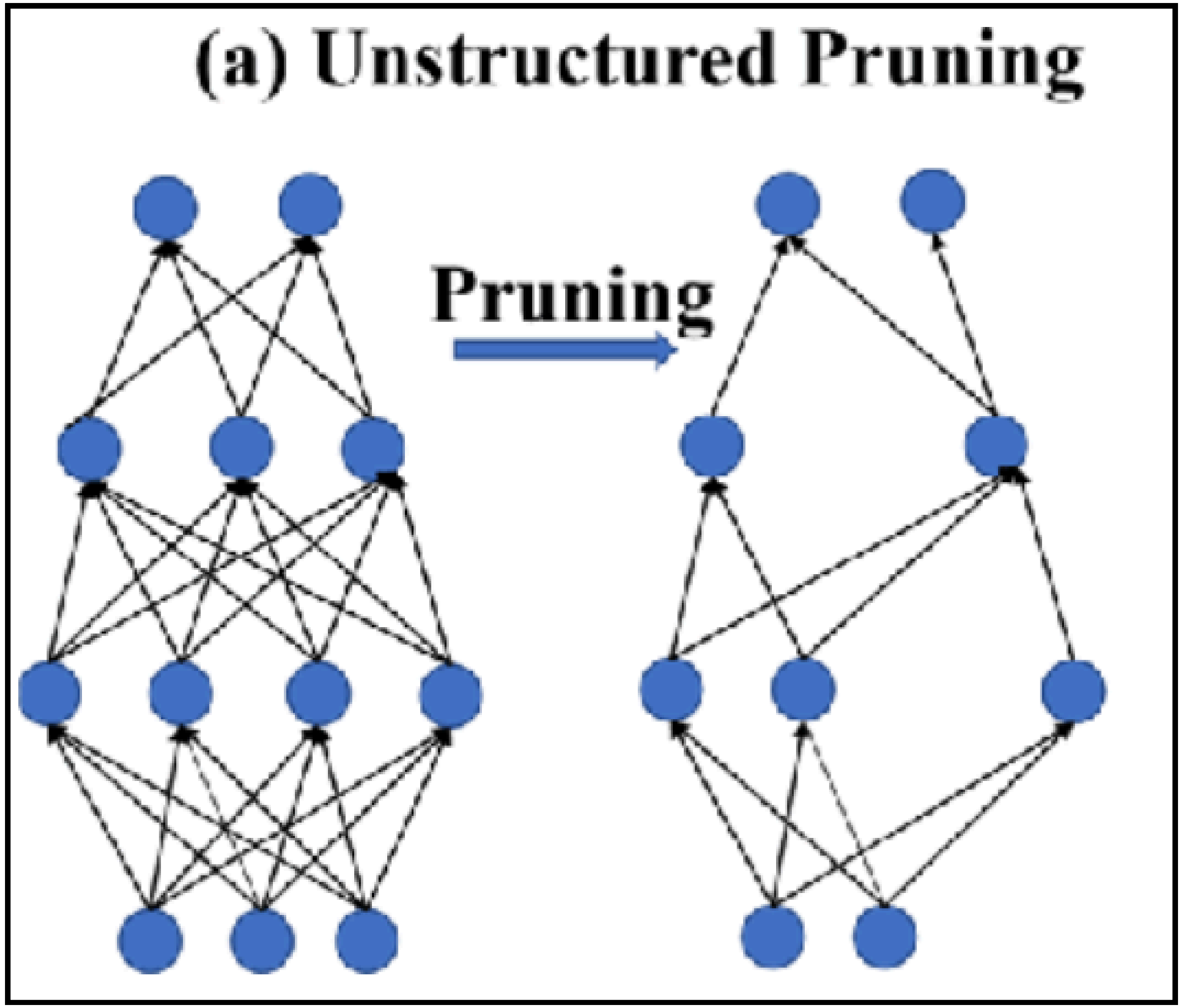
Other Models Used:

1) ResNet50
2) InceptionV3
3) InceptionResNetV2
4) DenseNet169

Ablation Table (CCE)					
Sl. No	Architecture	Performance Metrics			
		F1	Accuracy	Precision	Recall
1	ResNet50	0.966	0.9617	0.9637	0.9711
2	**DenseNet201**	**0.9819**	**0.9836**	**0.9811**	**0.983**
3	InceptionV3	0.954	0.97	0.9614	0.9479
4	InceptionResNetV2	0.9763	0.9807	0.9789	0.9738
5	DenseNet169	0.9858	0.9864	0.9877	0.9841

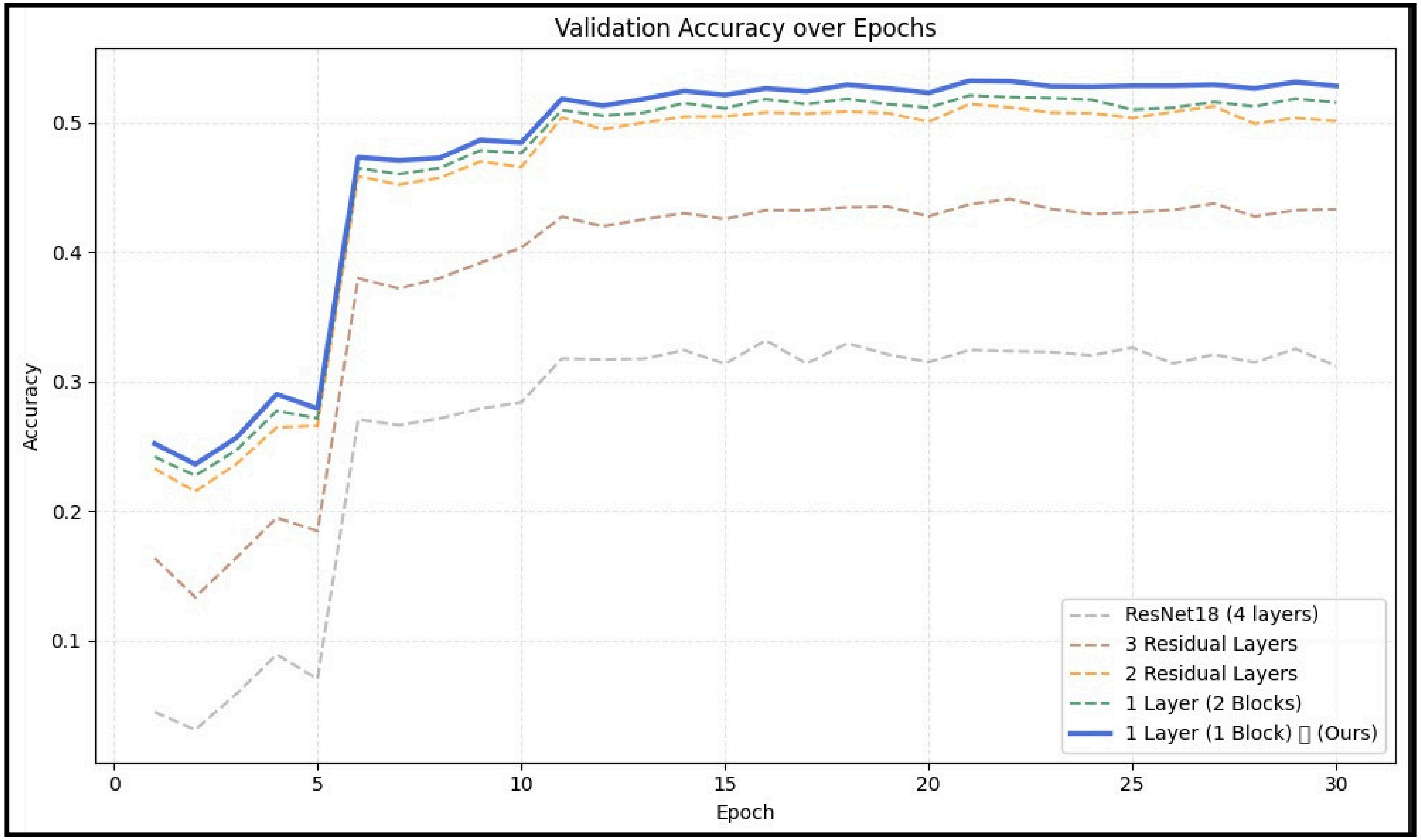
Ablation Table (SupCon)					
Sl. No	Architecture	Performance Metrics			
		F1	Accuracy	Precision	Recall
1	ResNet50	0.9856	0.9617	0.9637	0.9711
2	**DenseNet201**	**0.9948**	**0.9936**	**0.9944**	**0.9952**
3	InceptionV3	0.9732	0.9857	0.974	0.9724
4	InceptionResNetV2	0.9824	0.9833	0.9889	0.9838
5	DenseNet169	0.9902	0.9891	0.9931	0.9917

L1 Unstructured Weighted Pruning

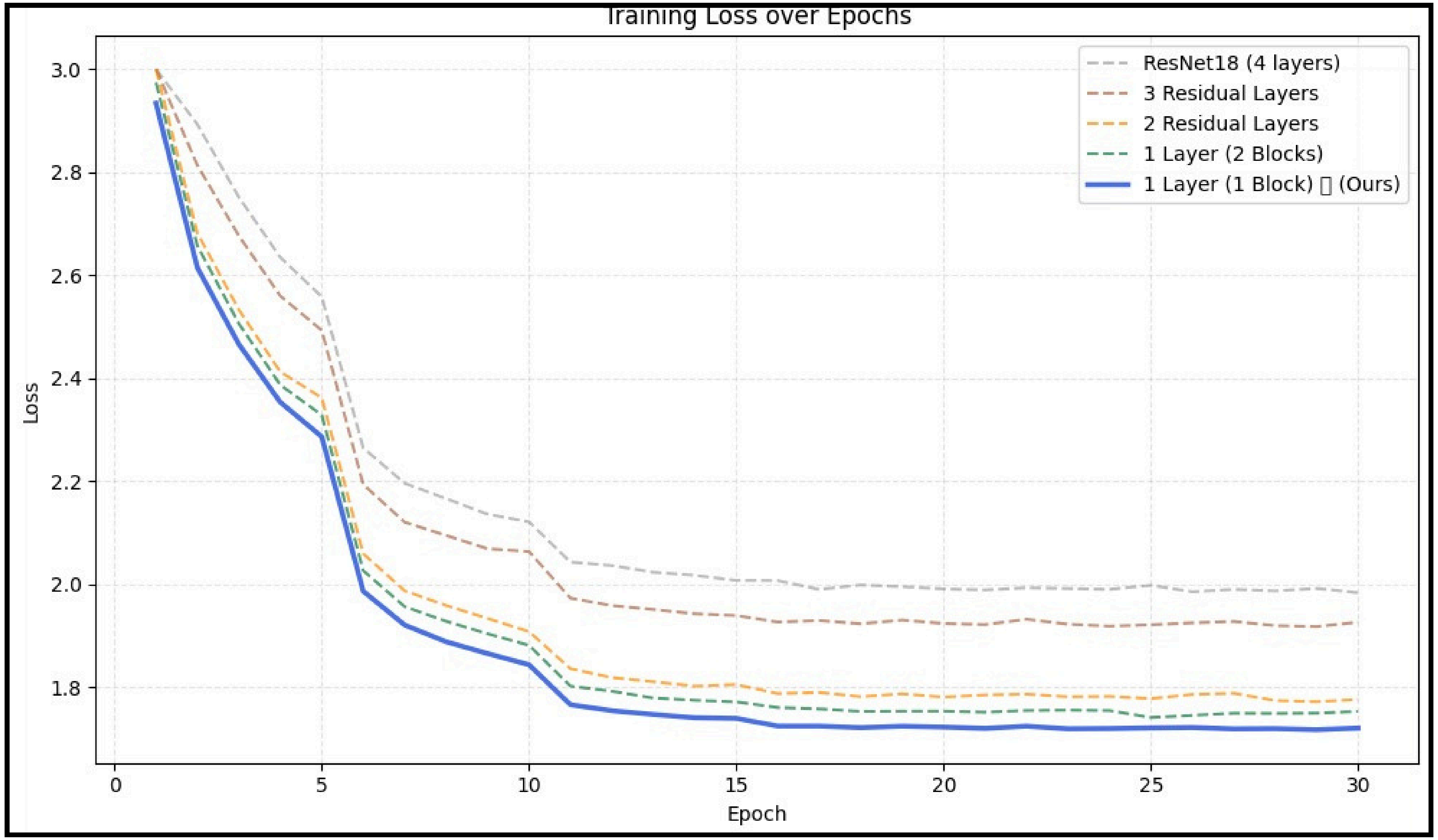


Loterry Ticket Hypothesis Style Pruning - Post Training Pruning

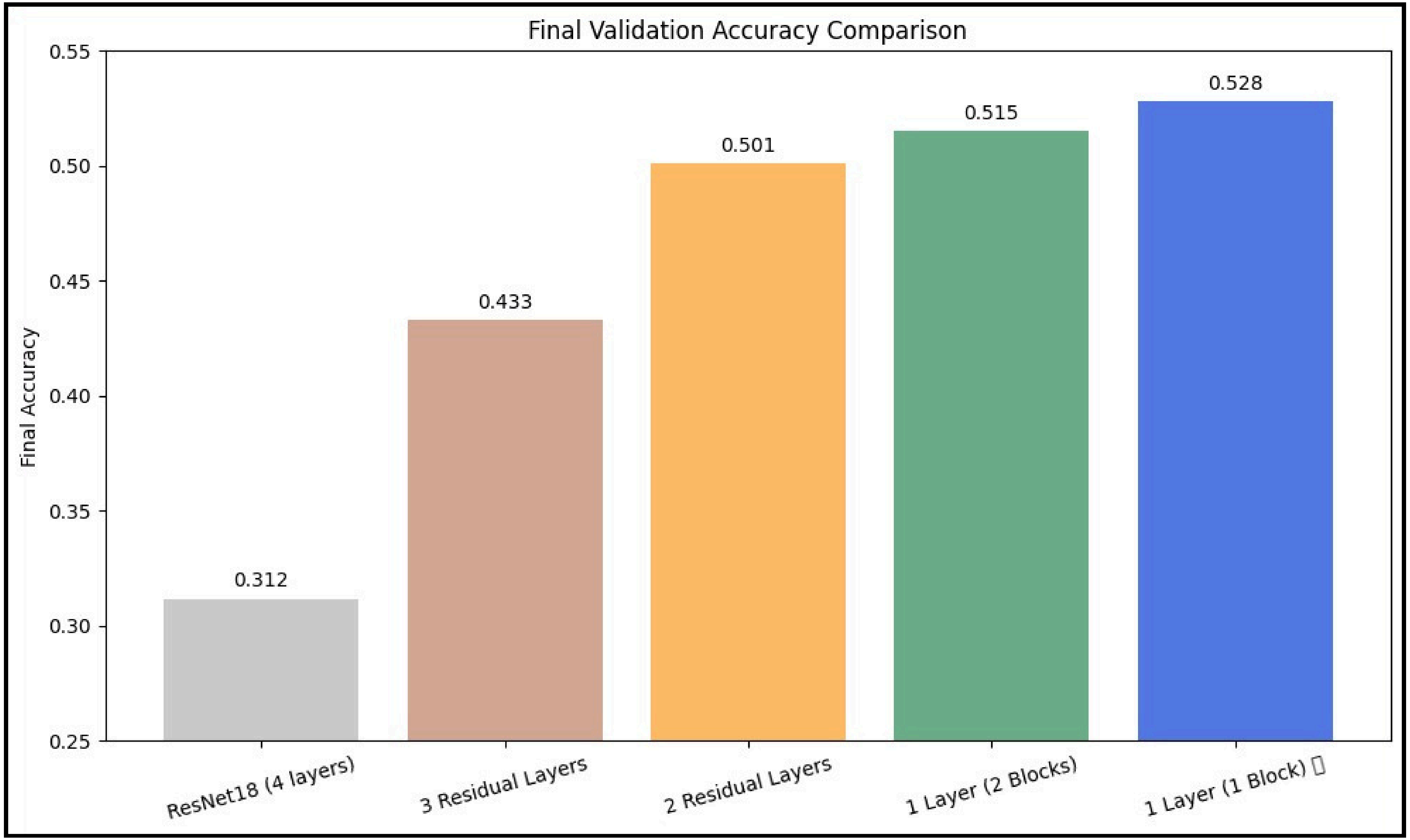
Comparison of Final Results



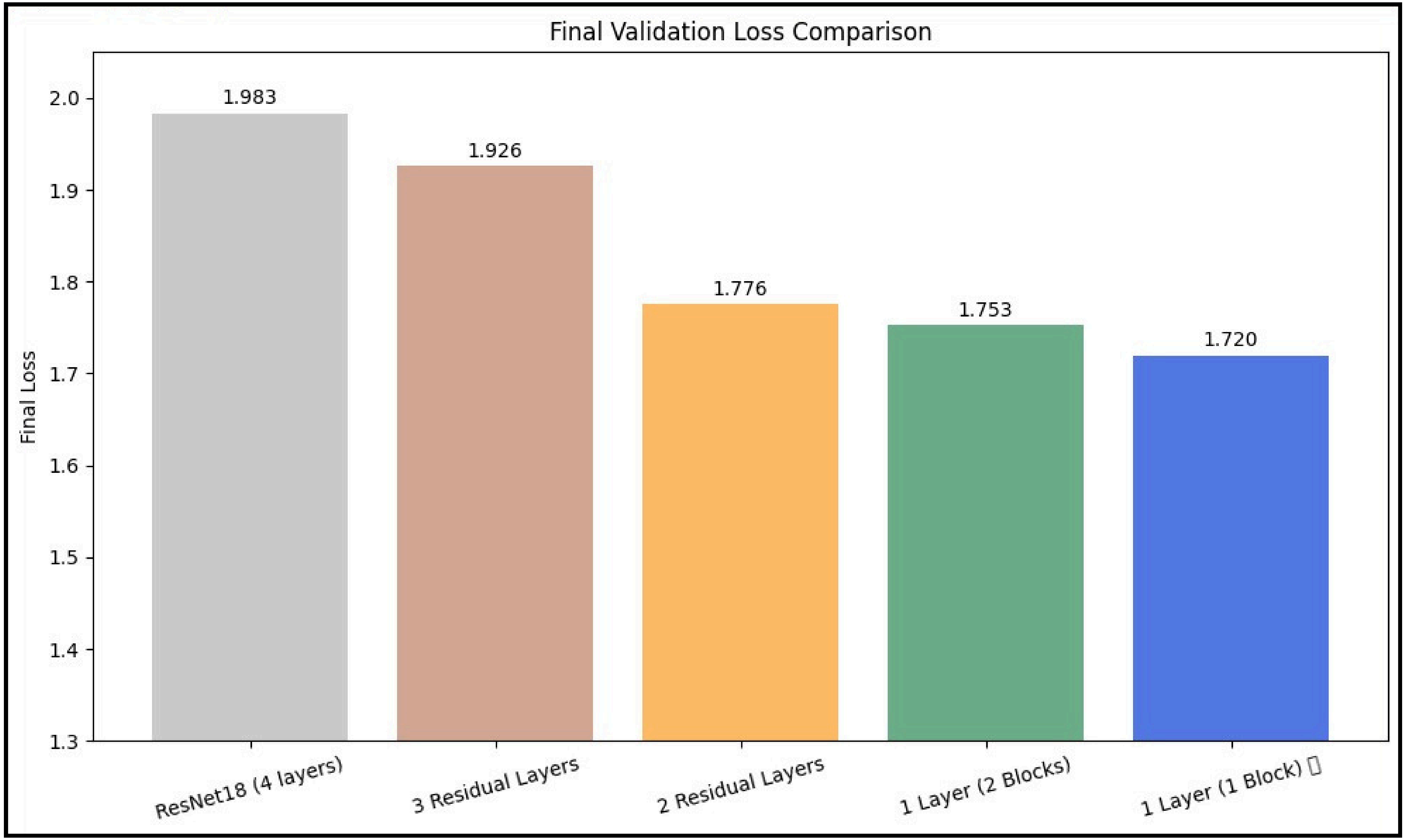
Comparison of Final Results



Final Validation Accuracy Comparison



Final Validation Accuracy Comparison

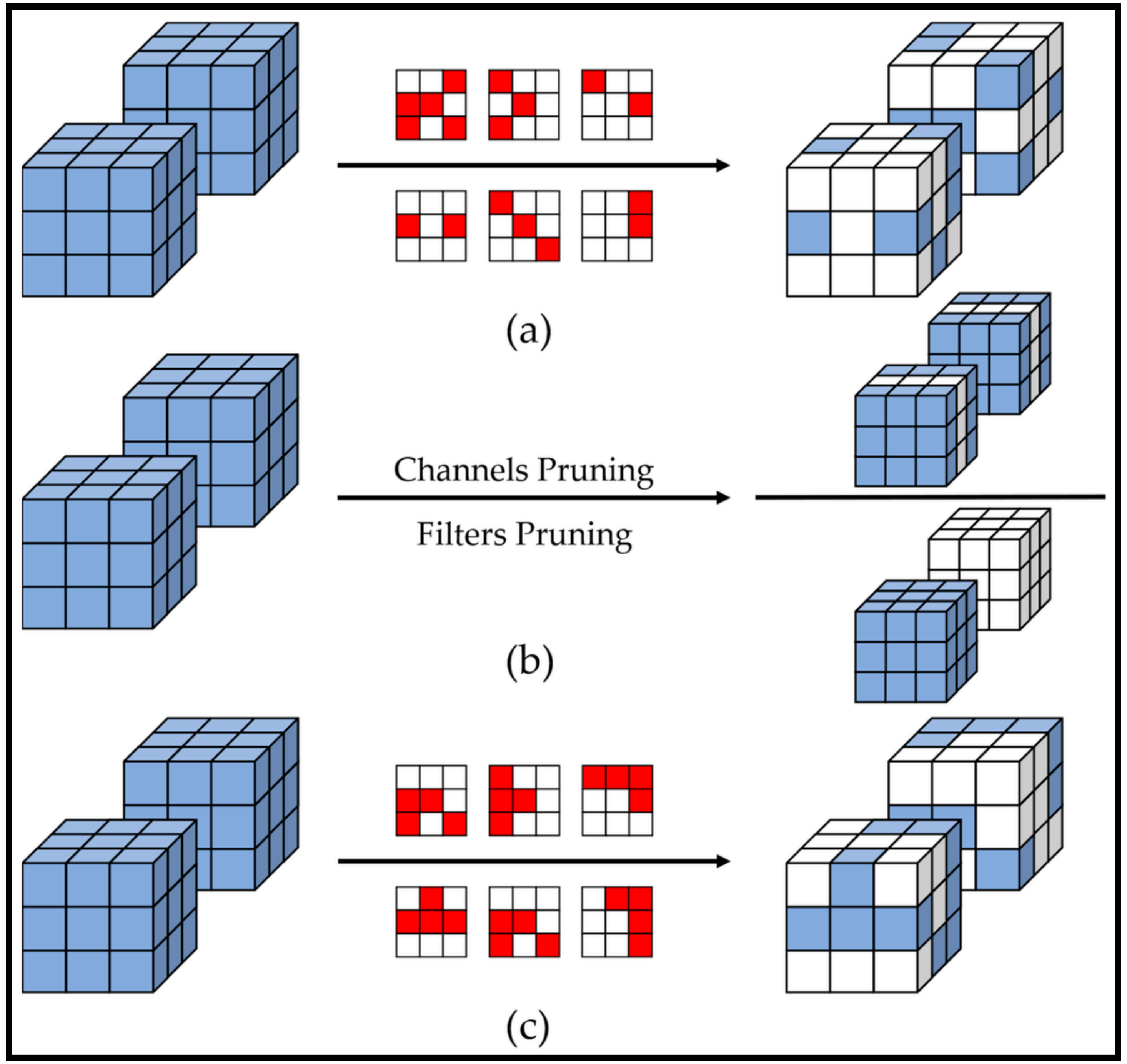


Why 50% Pruning?

Pruning Rate	Model Size	Compression Ratio	Accuracy
0% (baseline)	11.2MB	1.0x	54%
20%	9.1MB	1.23x	53.80%
40%	3.4MB	3.29x	53.42%
50%	0.358MB	31.3x	53.21%

Layer Sensitive Pruning

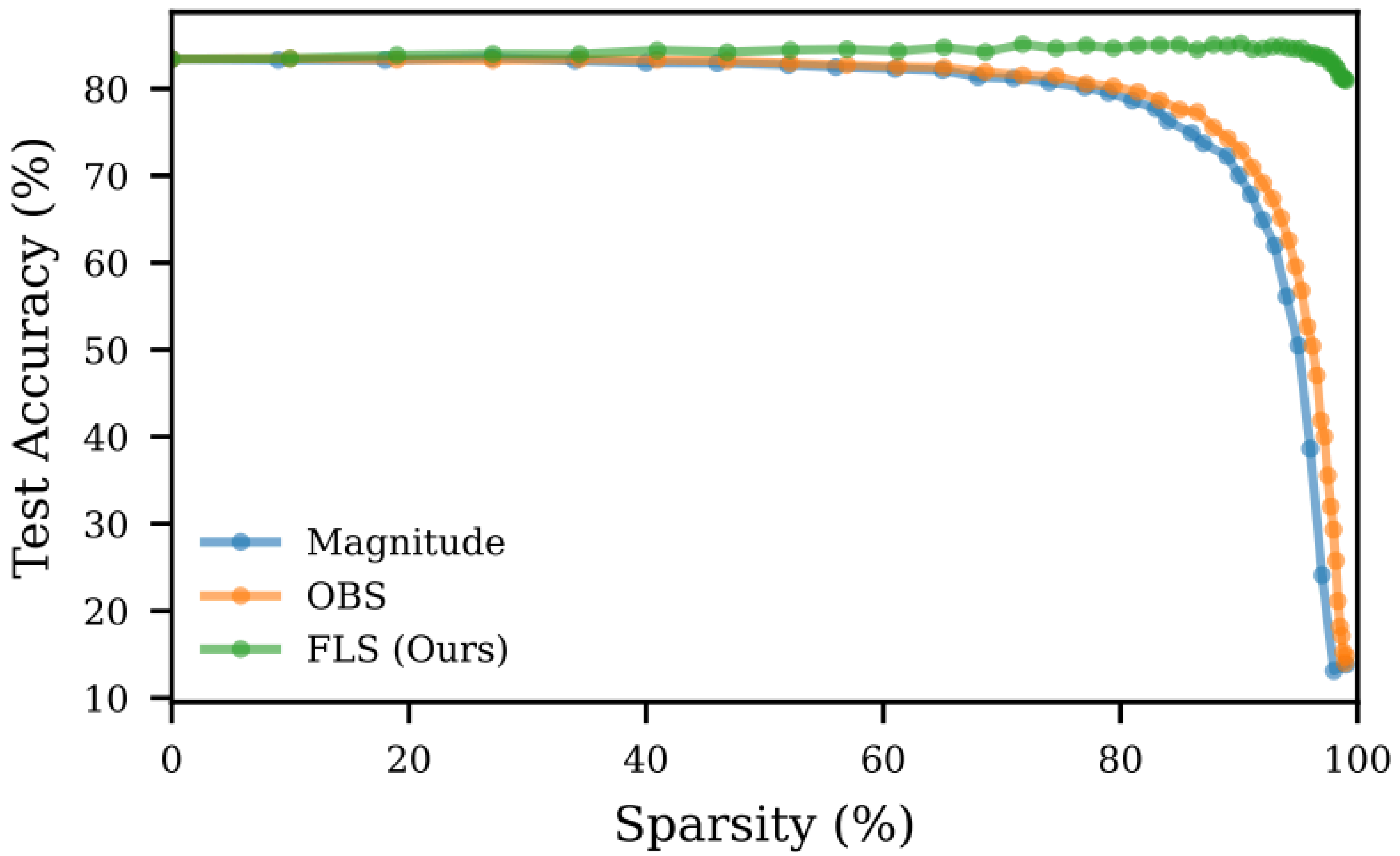
- Preserves Critical Features: Sensitive layers retain important weights, preventing accuracy drop.
- Balances Sparsity and Utility: Adapts pruning aggressiveness per layer, avoiding over- or under-pruning.
- Improves Generalization: Pruning less in high-impact layers reduces overfitting risk.
- Boosts Compression Efficiency: Achieves higher overall sparsity with minimal performance loss.



Scope Of Improvement

FishLeg Pruning

- FLS builds on the FishLeg optimizer to directly learn the inverse Fisher Information Matrix (FIM) using a meta-learned parametric form Q(λ).
- FLS uses second-order information (curvature of the loss surface) to assess true impact of removing each weight — leading to much more informed pruning.
- OBS-like second-order methods are accurate but slow — they require recomputing the Fisher matrix or Hessian repeatedly.

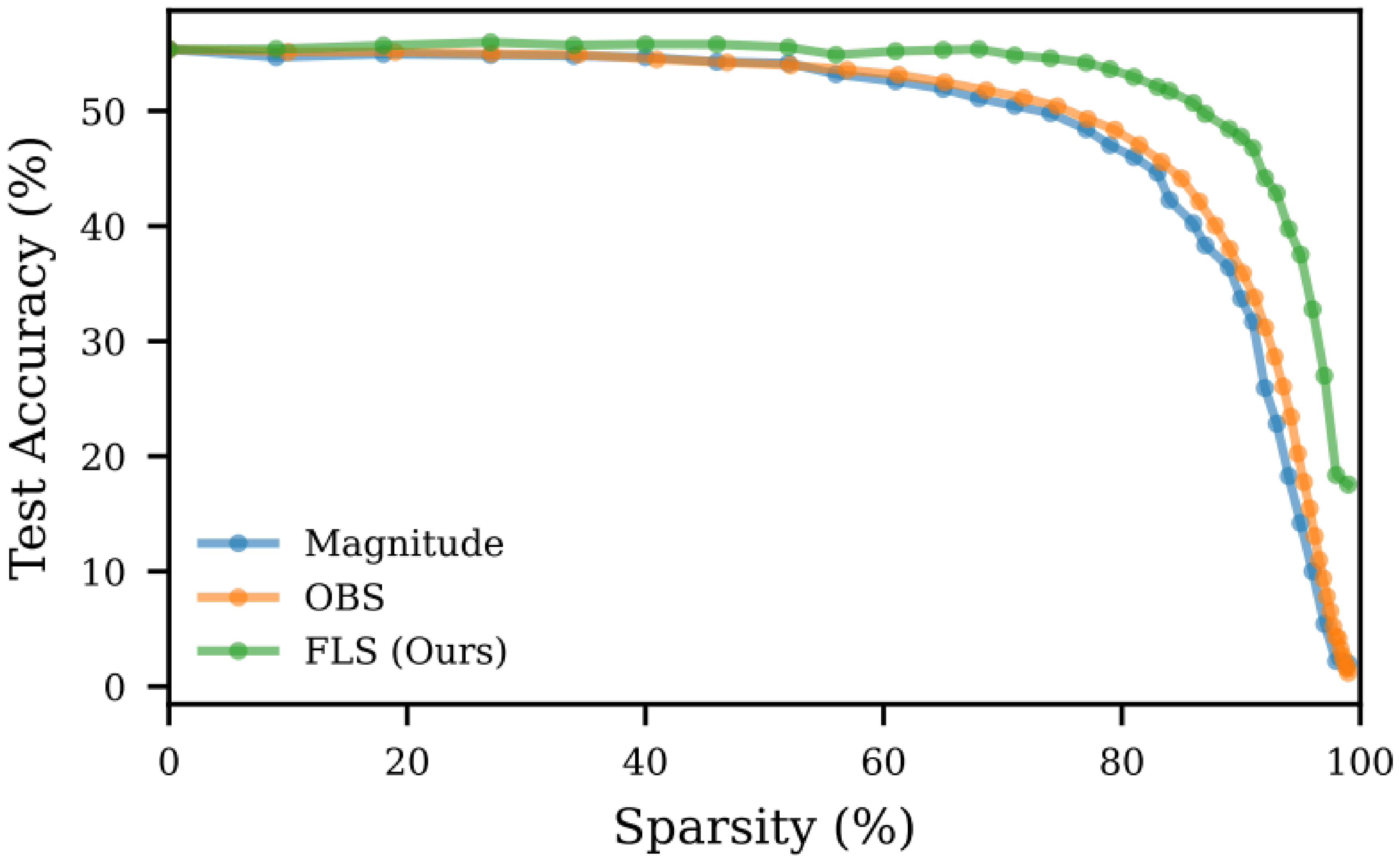


McGowan, J., Lai, W. S., Chen, W., Aldridge, H., Clarke, J., Garcia, J., Xia, R., Liang, Y., Hennequin, G., & Bernacchia, A. (2024). Efficient Model Compression Techniques with FishLeg. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2412.02328

Scope Of Improvement

FishLeg Pruning

- FLS avoids this by learning a compact, structured representation of the inverse Fisher — so it saves time and memory.
- Supports unstructured and semi-structured (2:4) pruning.
- Utilizes tensor factorization (Kronecker/block-diagonal) for memory efficiency.
- Uses smart initialization ($\alpha \approx \gamma^{-1}$) and preconditioning for faster convergence.



McGowan, J., Lai, W. S., Chen, W., Aldridge, H., Clarke, J., Garcia, J., Xia, R., Liang, Y., Hennequin, G., & Bernacchia, A. (2024). Efficient Model Compression Techniques with FishLeg. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2412.02328