IMAGE SIMILARITY

- Arvind Nair
- Landon Spitzer
- Tenzin Shenyen
- Sanskruti Mali

Project Overview

- Task: Given two images from the TinyImageNet dataset, predict whether the images are similar or dissimilar.
- Similar Images: If both images belong to the same class category among the 200 available classes.
- Dissimilar Images: If the two images belong to different class categories.
- Objective: Build a model that can automatically classify image pairs as similar or dissimilar based on visual features.
- Two deep learning methods applied for this task:
    - MatchNet: A patch-based metric learning approach.
    - BEiT: A transformer-based image representation method leveraging masked image modeling.

METHODS

MatchNet

- Patch-based matching method
- Learns feature representations + comparison function
- Siamese-style network architecture

BEiT

- BEiT: "BERT for Images"
- Transformer-based architecture
- Pretrained on large image datasets
- Learns token representations for patches

PAPER OVERVIEW- BEiT

Goal: Self-supervised pretraining for Vision Transformers using ideas from BERT.

Masked Image Modeling (MIM):
- Randomly mask image patches.
- Predict corresponding *visual tokens* (not raw pixels) using Transformer.

Image Representation:
- Input: Image patches (16×16 pixels each).
- -Output: Visual tokens generated by a discrete VAE tokenizer.

Backbone Network:
- Standard Vision Transformer (ViT) encoder.
- No task-specific heads during pretraining.

Training Strategy:
- Mask about 40% of patches.
- Predict missing visual tokens using a softmax classifier.

Key Innovations:
- Predicting discrete visual tokens helps focus on semantic content instead of fine pixel details.
- Inspired by Variational Autoencoder theory (VAE + masked token recovery).

Key Results:
- Strong performance on ImageNet classification and semantic segmentation.
- Learned attention maps show *emerging semantic structure* without supervision.

Takeaways:
- BERT-style masked modeling works very well for images when done at the token level instead of raw pixels.

PAPER OVERVIEW - MatchNet

Goal: Jointly learn *both* feature extraction and similarity metric for patch-based image matching.

Architecture:

- **Feature Network**: Deep CNN to encode patches into compact representations (inspired by AlexNet).
- **Metric Network**: Three fully-connected layers to learn a *similarity score* between two patch representations.

Two-Tower Structure:

- Two identical feature extractors share weights ("Siamese-style").
- Outputs concatenated and passed to metric network.

Training Strategy:

- Supervised with cross-entropy loss (predicting match / no-match).
- Data augmentation and balanced positive/negative sampling to prevent overfitting.

Efficiency:

- During inference, feature extraction and matching run separately (feature caching).

Key Results:

- Outperforms SIFT and previous learned descriptors on UBC patch dataset.
- Achieves better accuracy with smaller descriptors (fewer bits).

Takeaways:

- Unified learning of representation **and** comparison is more powerful than handcrafting features or metrics separately.

Dataset and Preprocessing

Resizing Images

Resizing images to 64x64 pixels to match TinyImageNet input requirements

Normalization

Normalization of pixel values to the range [0, 1]

Forming pairs

Generating all possible ordered pairs of images from the test split

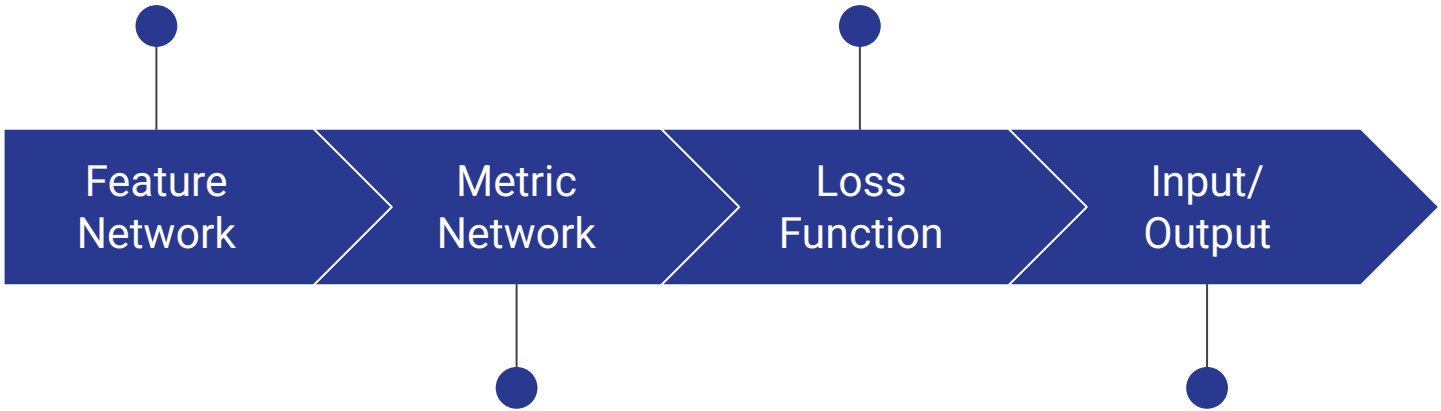
Labeling each pair: 1 (similar) if both images belong to the same class, 0 (dissimilar) otherwise

Implementation

MatchNet Architecture

Typical layers:
Convolution + ReLU +
Pooling blocks.

Binary Cross-Entropy
Loss.



Computes a similarity score between the two feature vectors.

- Input: A pair of images (Image A, Image B).
- Output: Probability score indicating whether the images are similar.

Model Training and Validation Overview (MatchNet)

Model:

- Custom MatchNet model for image similarity learning.
- Feature extractor + bottleneck + metric network structure.

Training Details:

- Dataset split into Training, Validation, and Testing sets.
- Training optimizes Binary Cross Entropy Loss (BCELoss).
- Validation monitors generalization performance after each epoch.

Hyperparameters Used:

- Learning Rates Tested: 0.0001, 0.0005, 0.001
- Batch Size: 32
- Bottleneck Dimension: 256
- Dropout Rate: 0.3
- Optimizer: Adam (Learning Rate Scheduling not used)
- Epochs: 5 per learning rate
- Image Size: 64x64

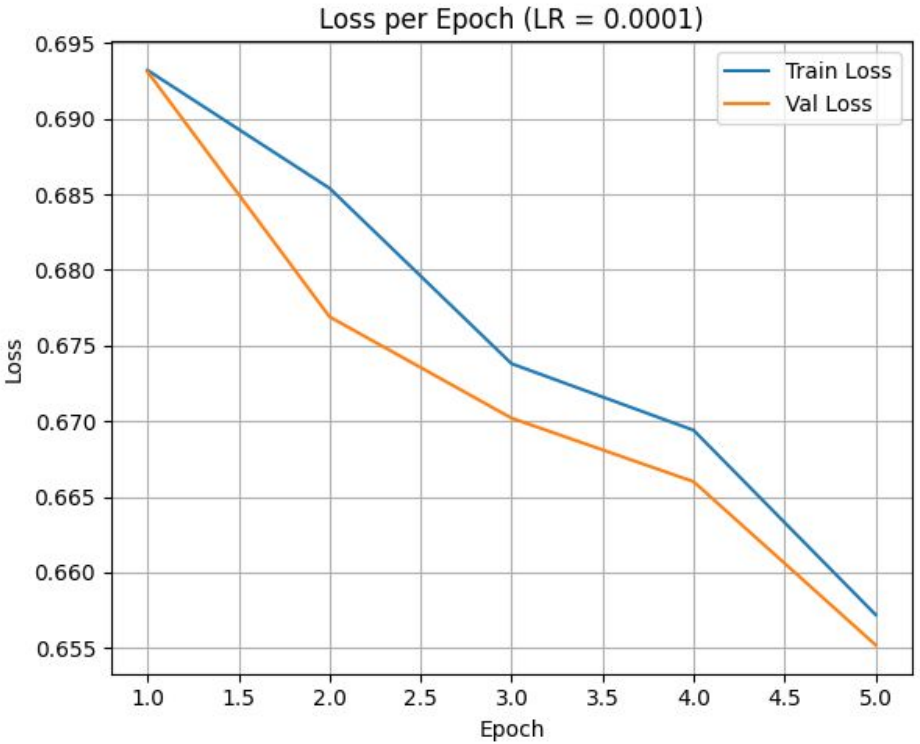
Evaluation Strategy:

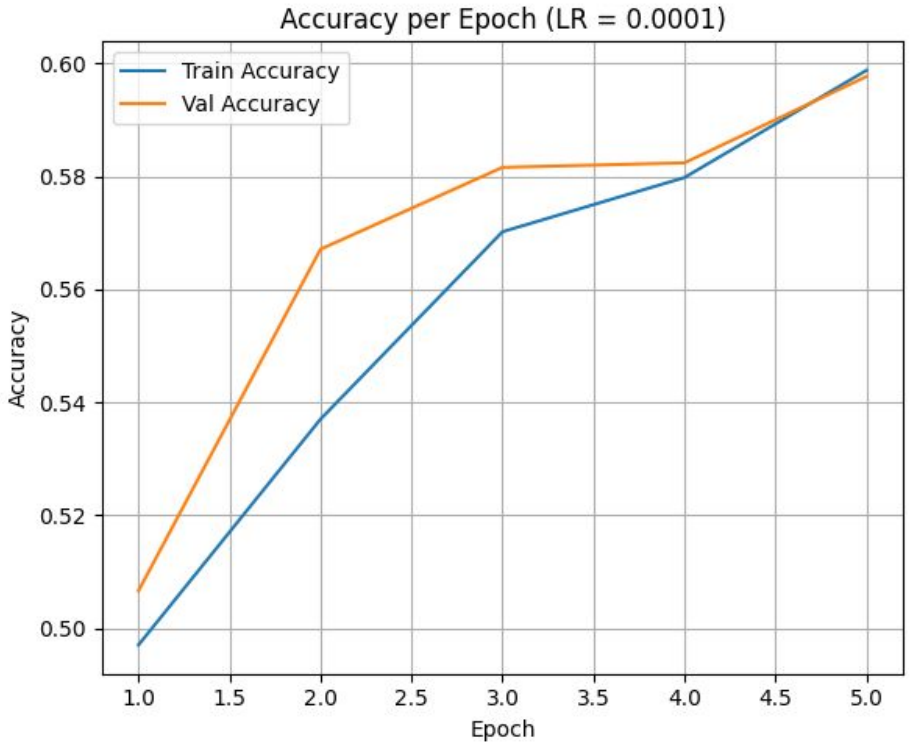
- Validation accuracy measured after each epoch.
- ROC Curve and AUC calculated for model performance.
- Final models saved after training for separate testing.

Training and Validation Metrics Across Learning Rates

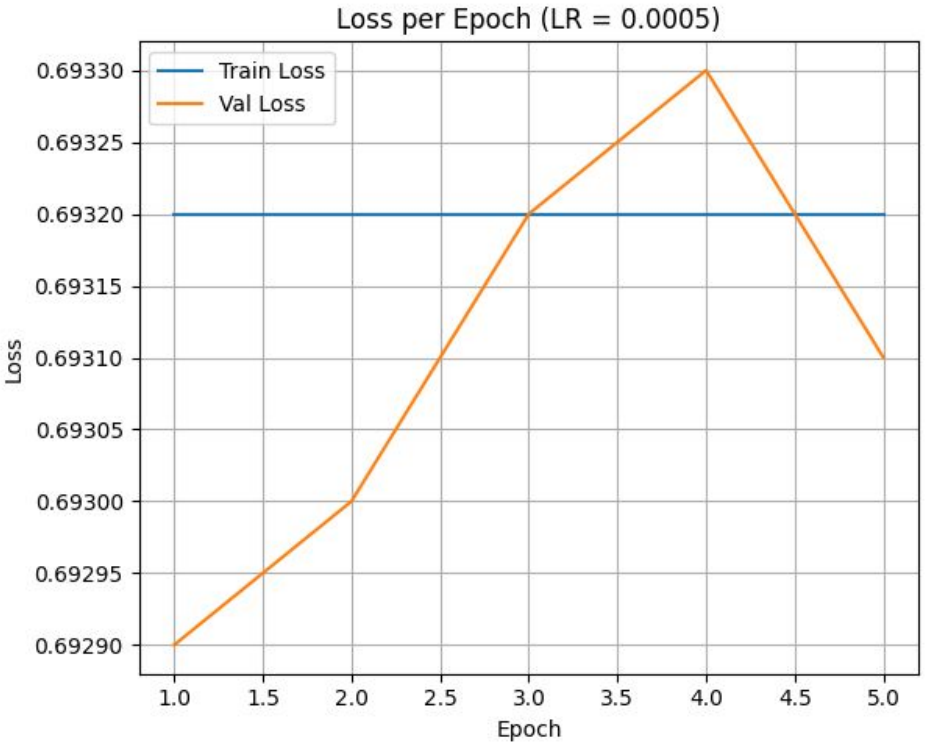
Learning Rate	Epoch	Train Loss	Train Accuracy	Val Loss	Val Accuracy
0.0001	1	0.6932	0.497	0.6931	0.5066
	2	0.6854	0.537	0.6769	0.5671
	3	0.6738	0.5702	0.6702	0.5816
	4	0.6694	0.5798	0.666	0.5824
	5	0.6572	0.5988	0.6552	0.5977
0.0005	1	0.6932	0.5012	0.6929	0.5129
	2	0.6932	0.4988	0.693	0.5093
	3	0.6932	0.5016	0.6932	0.4958
	4	0.6932	0.4986	0.6933	0.4954
	5	0.6932	0.5014	0.6931	0.5188
0.001	1	0.6934	0.5008	0.6932	0.4917
	2	0.6932	0.5	0.6932	0.4987
	3	0.6931	0.5029	0.693	0.5107
	4	0.6932	0.501	0.6931	0.518
	5	0.6932	0.5026	0.6932	0.4917

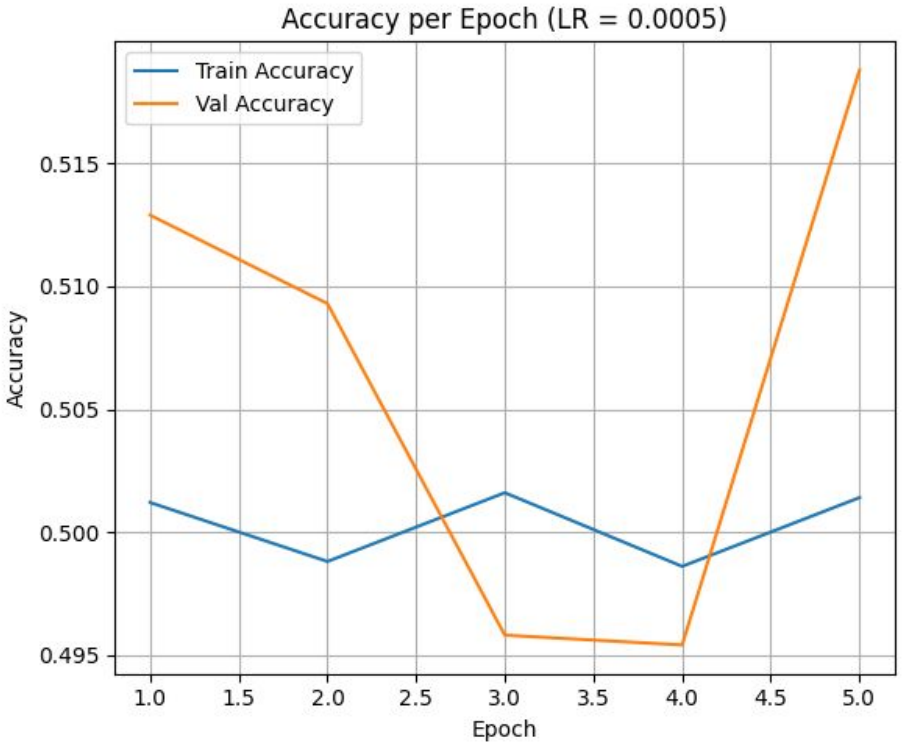
Training and Validation Curves (LR = 0.0001)



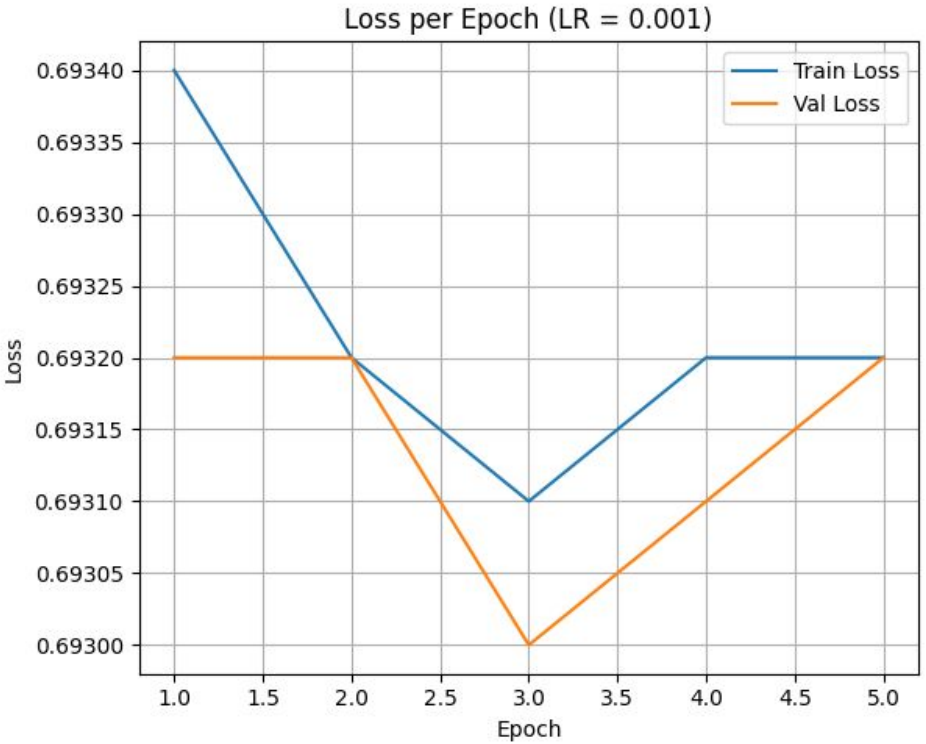


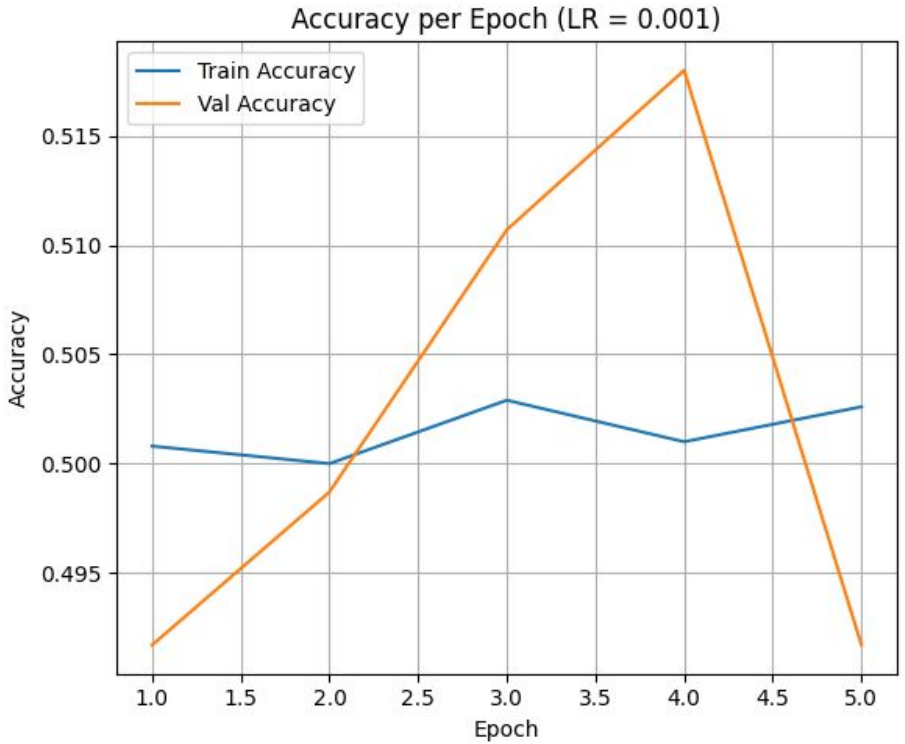
Training and Validation Curves (LR = 0.0005)



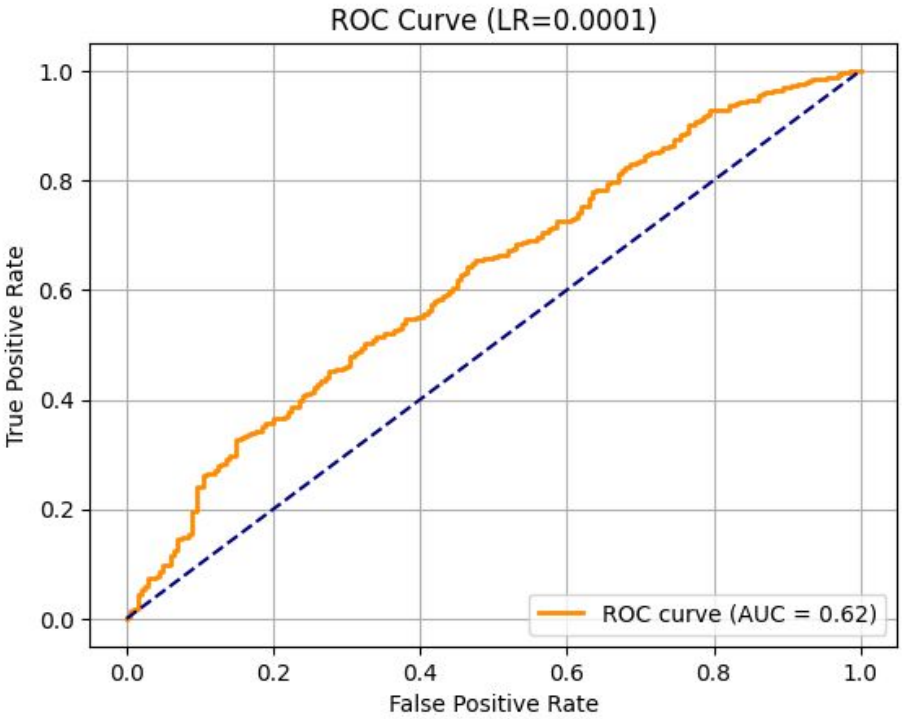


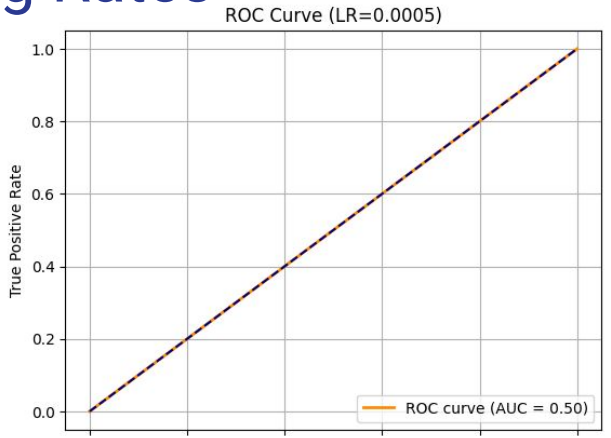
Training and Validation Curves (LR = 0.001)

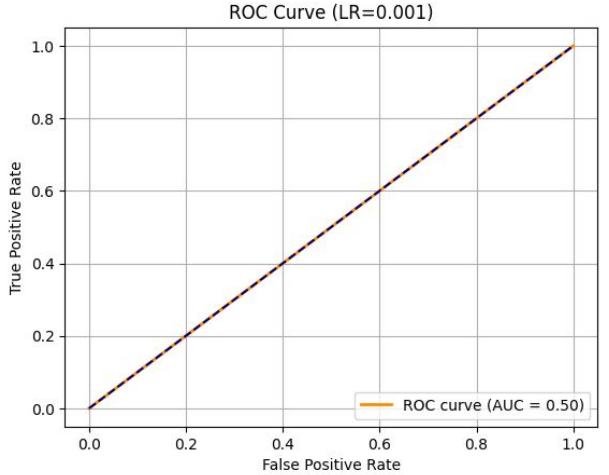




ROC Curves Across Different Learning Rates

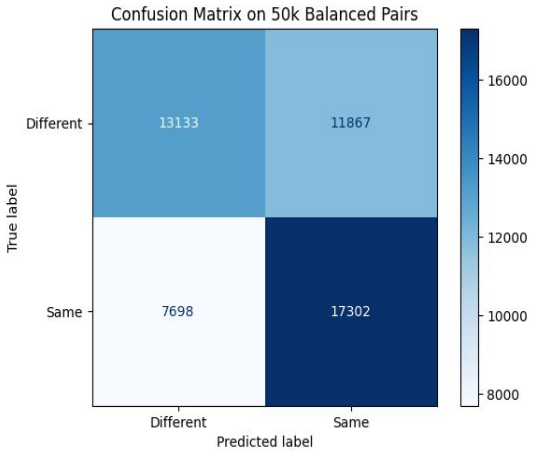




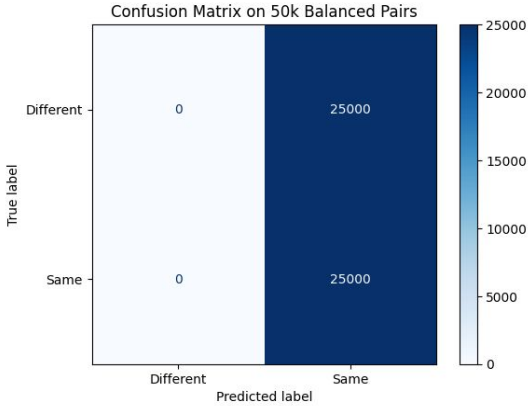


Confusion Matrices Across Learning Rates (50k Balanced Pairs)

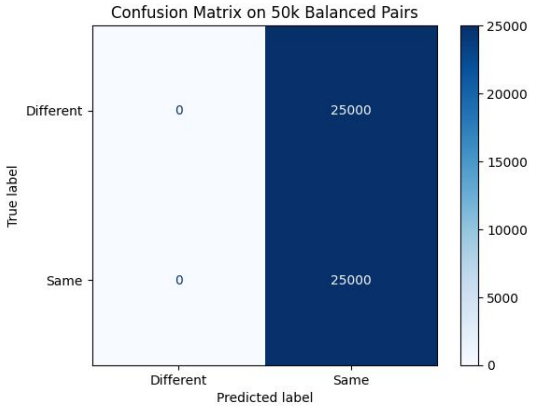
confusion matrices across different learning rates, based on 50,000 balanced pairs (25k "Same", 25k "Different")



Learning Rate = 0.0001
→ Final Accuracy: 60.87%



Learning Rate = 0.0005
→ Final Accuracy: 50.00%

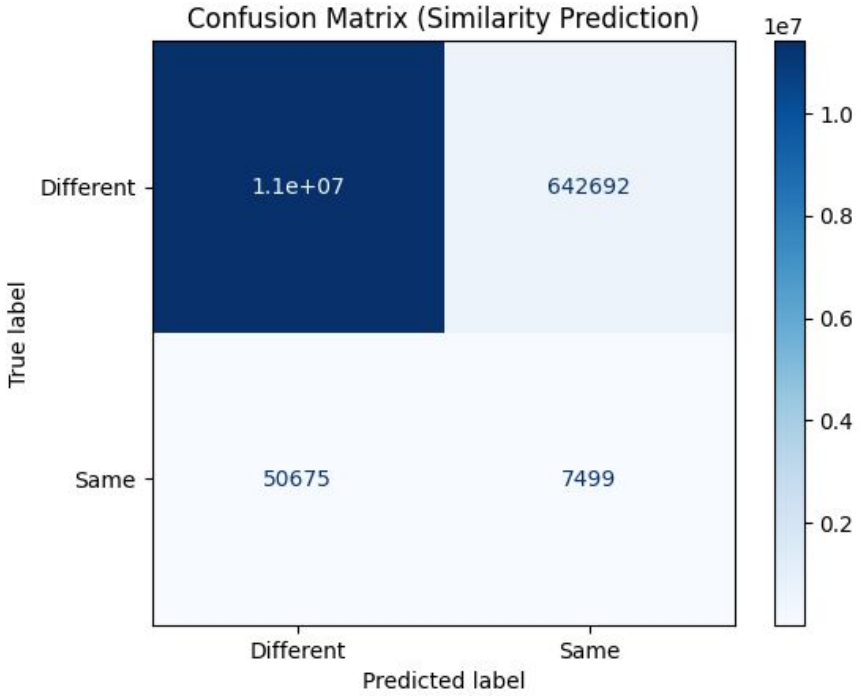


Learning Rate = 0.0005
→ Final Accuracy: 50.00%

Confusion Matrix Summary (Tested on 12,115,503 pairs using MatchNet)

- True Label: Different, Predicted: Different → 11,000,000
- True Label: Different, Predicted: Same → 642,692
- True Label: Same, Predicted: Different → 50,675
- True Label: Same, Predicted: Same → 7,499

Final Similarity Prediction Accuracy: 94.28%



BEiT

The input image is divided into a grid of small, non-overlapping patches (e.g., 16×16 pixels).

This enables the model to learn rich contextual relationships between different patches across the entire image.

After pretraining, BEiT is fine-tuned on a pairwise similarity task.

Each patch is flattened into a 1D vector and then projected into a fixed-dimensional embedding space through a learnable linear layer

Learning to reconstruct missing parts forces the model to understand global image structure.

Model Training and Validation Overview (BEIT)

Model:

- BEiT, which stands for Bidirectional Encoder representation from Image Transformers, adapts the masked language modeling strategy of BERT to the visual domain. Instead of predicting masked words, BEiT predicts masked visual tokens derived from images.

Training Details:

- Dataset split into Training, Validation, and Testing sets.
- Training optimizes Binary Cross Entropy Loss (BCELoss).
- Validation monitors generalization performance after each epoch.

Hyperparameters Used:

- Learning Rates Tested: 0.00002, 0.00005
- Batch Size: 32, 64, 128
- Bottleneck Dimension: 256
- Weight Decay: 0.01, 0.05
- Optimizer: Adam (Learning Rate Scheduling not used)
- Epochs: 5 per learning rate
- Image Size: 64x64

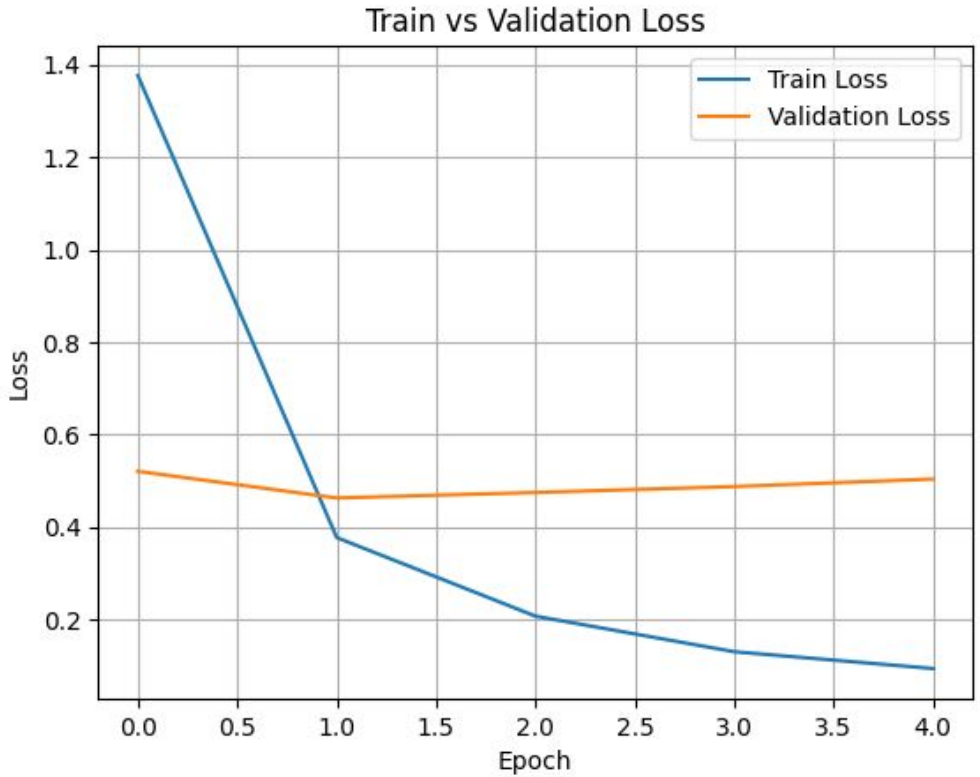
Evaluation Strategy:

- Training & Validation loss & accuracy measured after each epoch.
- Final models saved after training for separate testing.

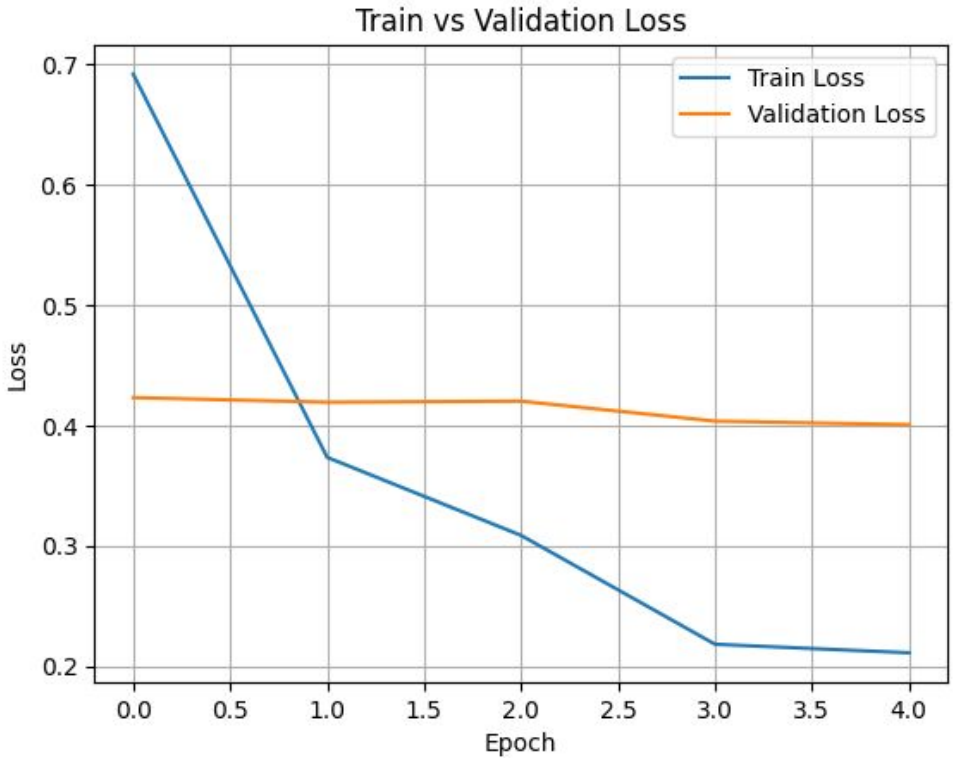
Training and Validation Metrics Across Learning Rates

Learning Rate / Batch Size	Epoch	Train Loss	Train Accuracy	Val Loss	Val Accuracy
2e-5 / 32	1	4.5340	0.0794	3.9627	0.1495
	2	3.6231	0.1966	3.4701	0.2198
	3	3.1831	0.2710	3.0167	0.3060
	4	2.8463	0.3301	2.8761	0.3255
	5	2.5549	0.3839	2.6836	0.3585
5e-5 / 64	1	0.6920	0.8455	0.4231	0.8878
	2	0.3734	0.8973	0.4192	0.8878
	3	0.3085	0.9120	0.4201	0.8886
	4	0.2180	0.9377	0.4035	0.8918
	5	0.2109	0.9405	0.4006	0.8916
5e-5 / 128	1	2.7740	0.5959	1.0456	0.8503
	2	0.8366	0.8503	0.6006	0.8729
	3	0.5993	0.8670	0.5056	0.8796
	4	0.5380	0.8738	0.4992	0.8796
	5	0.5280	0.8766	0.4932	0.8806

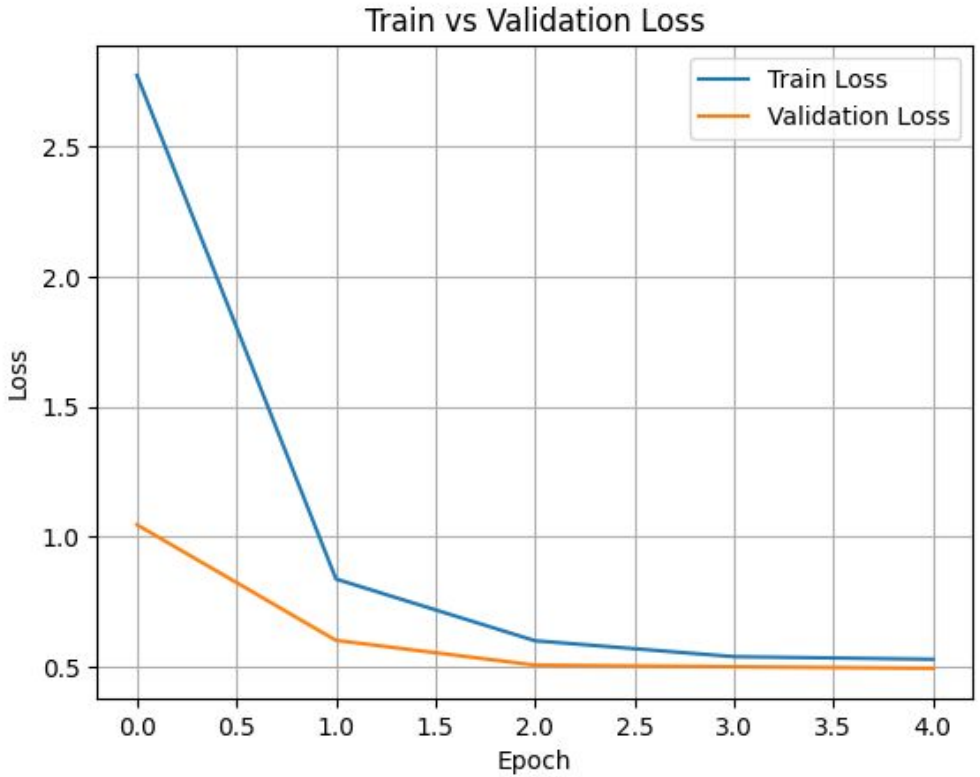
Training and Validation Curves (LR = 2e-5, BS = 32)



Training and Validation Curves (LR = 5e-5, BS = 64)



Training and Validation Curves (LR = 5e-5, BS = 128)

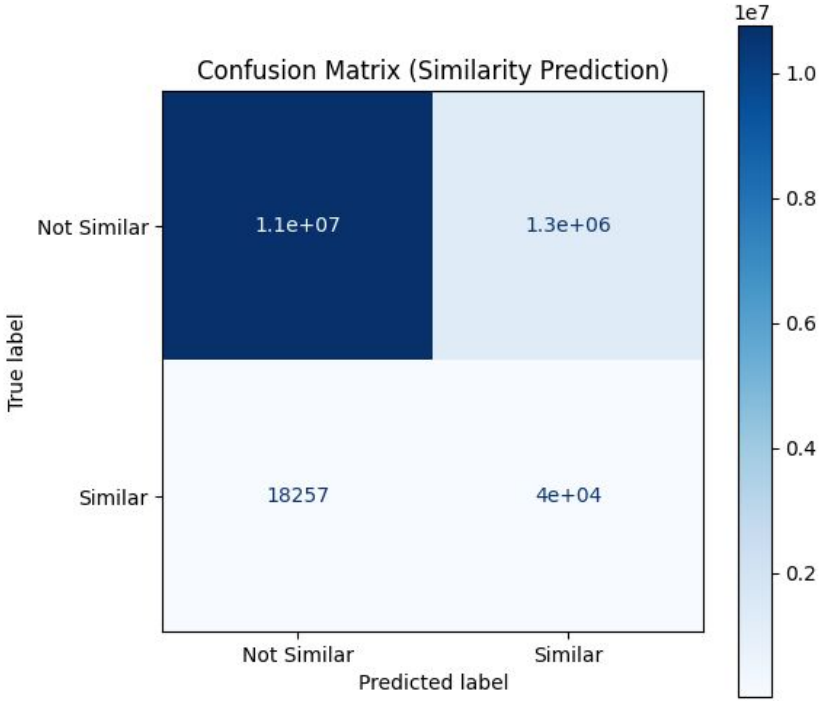


Confusion Matrix Summary (Tested on 12,115,503 pairs using BEIT)

The best model we found out was with learning rate as 5e-5 and the batch size as 64

- True Label: Different, Predicted: Different → 10,759,967
- True Label: Different, Predicted: Same → 1,297,362
- True Label: Same, Predicted: Different → 18,257
- True Label: Same, Predicted: Same → 39,917

Final Similarity Prediction Accuracy: 89.14%



Thankyou