# ATML: PA0

**Sarim Malik; 26100129**

## Project Repository

github.com/s-malix21/atml-fall2025

## 1. Introduction

## 2. TASK 1

### 2.1. Baseline Setup

#### 2.1.1. Why not train ResNet-152 from scratch on small datasets?

ResNet-152 has a high num of params. Small datasets such as CIFAR-10 do not have enough scale or diversity, which results in severe overfitting and poor generalization when trained from scratch. Convergence is also extremely slow. Pretraining on ImageNet or another large-scale dataset provides a much stronger starting point, since the learned features are transferable.

#### 2.1.2. Freezing backbone: what does this say about transferability?

When only the final layer was trained and the rest of the backbone frozen, the model still reached a validation accuracy of about 83%. This clearly shows that early and mid-level layers capture generic features (edges, textures, shapes) which are highly transferable across visual tasks. Transfer learning in this case works very effectively.

### 2.2. Residual Connections in Practice

#### 2.2.1. Effect on gradient flow

Skip connections allow gradients to bypass certain layers, which prevents vanishing gradients and makes optimization feasible even in extremely deep models. Without these shortcuts, training such networks becomes impractical.

#### 2.2.2. Removing residuals: convergence and performance

Once residuals were disabled, validation accuracy collapsed to roughly 22% compared to the 83% baseline. Convergence slowed dramatically and optimization became unstable. This confirms that residual connections are absolutely essential for both faster convergence and final model performance.

### 2.3. Feature Hierarchies and Representations

#### 2.3.1. How class separability evolves across layers

t-SNE and UMAP visualizations showed that in the early layers, class clusters were highly overlapping, indicating poor separability. As the network depth increased, clusters corresponding to different classes became more distinct. The representations gradually transformed from raw pixel information into discriminative class-level features.

#### 2.3.2. Low-level vs. high-level features

Low-level representations were dominated by edges and color patterns, which led to mixed and noisy clusters. High-level features, in contrast, encoded semantic information that resulted in clear separation and clean clustering in the embedding space. This progression highlights the hierarchical nature of learned features.

### 2.4. Transfer Learning and Generalization

#### 2.4.1. Best trade-off between compute and accuracy

Fine-tuning only the final block gave a validation accuracy of around 72%. This offered the best balance, since it was significantly more efficient than full fine-tuning while still delivering strong performance. Freezing everything except the last layer was faster but less accurate, while full fine-tuning achieved the best accuracy but required the most compute.

#### 2.4.2. Which layers transfer best and why

The early and middle layers transferred most effectively because they encode general-purpose visual features. The final block tends to be task-specific, so fine-tuning it allows the model to adapt without discarding the benefits of pre-training. This makes fine-tuning the last block the most efficient strategy overall.

# 3. TASK 2

## 3.1. Using a Pre-trained ViT for Image Classification

### 3.1.1. Top-1 predictions

Image 1 predicted lorikeet, matched bird. Image 2 predicted airliner, correct. Image 3 predicted Granny Smith, correct for green apple. Image 4 predicted soccer ball, correct. All predictions matched the objects.

## 3.2. Visualizing Patch Attention

### 3.2.1. Attention map overlay

Overlays showed attention focused on relevant regions, e.g. bird patches in the first image. Patches highlighted lined up with the object.

## 3.3. Analyzing the Attention Map

### 3.3.1. Focus on predicted class regions

Attention concentrated on the object areas (bird, apple, ball).

### 3.3.2. Comparison to CNN attention

ViT attention distributed patch-wise, CNN Grad-CAM smoother and spatially contiguous. ViT attention weights available directly, no extra computation.

### 3.3.3. Head specialization and odd behaviors

With heavy or center masking, attention sometimes shifted to background and predictions failed. Head-wise maps showed some heads focusing on boundaries, others on object parts or background.

## 3.4. Patch Masking Analysis

### 3.4.1. Robustness to missing patches

ViT handled random masking better, still predicted lorikeet though with lower confidence. Center masking broke predictions since core object features were missing. Random masking less harmful due to distributed representation.

## 3.5. Pooling Method Comparison

### 3.5.1. CLS token vs mean pooling

CLS token pooling accuracy $0.9630$, mean pooling $0.9259$. CLS token better since trained to gather global info. Mean pooling diluted discriminative features.

### 3.5.2. Effect of pretraining objective

CLS token worked best here because pretraining used it for classification. Mean pooling may work better when pretraining is patch-level, but not in this case.

# 4. TASK 3

## 4.1. Baseline GAN Training

### 4.1.1. Loss curves and training dynamics

Discriminator loss started high and dropped, generator loss went up then both leveled out. Some fluctuations in the curves but no major oscillations.

### 4.1.2. Quality of generated images

Generator produced digits that were recognizable as MNIST, some variety across different noise inputs. Outputs change with different $z$, different digits and writing styles appear.

### 4.1.3. Comparison to reference implementations

Digits from this MLP GAN looked less sharp and less diverse compared to CNN-based GANs. Likely due to architecture choice (MLP vs conv) and hyperparameters like LR, batch size. Random initialization also plays a role.

## 4.2. Training Issues

### 4.2.1. Gradient vanishing

When discriminator was too strong (higher LR, more steps), accuracy saturated near 100% with $D(real) \approx 1.0$, $D(fake) \approx 1.0$. Generator loss stalled around $-100.0$, no improvement after that. Adding label smoothing and using non-saturating loss kept generator moving forward, discriminator scores stopped saturating. Gradients for G stayed informative.

### 4.2.2. Mode collapse

With stronger generator, samples collapsed into identical outputs. Diversity metrics dropped to zero (variance $= 0.000000$, mean distance $= 0.0000$). After balancing with stronger D and label smoothing, variance went back up ($0.260795$) and mean distance $20.1521$. Outputs showed variety again. Collapse happens when G finds one output that always fools D but ignores rest of distribution.

### 4.2.3. Discriminator overfitting

Large discriminator with limited data memorized training set. $D(real) = 0.9453$, test accuracy $= 1.000$, mean $D(real\_test) = 0.985$. Generator outputs looked worse, less diverse. Adding dropout and label smoothing lowered mean test score to $0.932$, curves became more stable, generated digits improved. Overfit D hurts G by giving bad gradients. Regularization makes D generalize better, G benefits.

# TASK 4

### 4.2 Visualize Reconstructions and Generations

Reconstructions match originals, slightly blurrier. Fine details lost, edges smooth. Class features (boots, shirts, coats) preserved. Generations from random latent vectors show FashionMNIST items. Shapes recognizable (shirts, shoes, bags). Less sharp, more generic. Diversity present but some samples blurry or ambiguous.

### 4.3 Posterior Collapse Investigation

Reconstruction loss drops fast then plateaus (32.0 $\rightarrow$ 14.1). KL rises early then stabilizes (5.8 $\rightarrow$ 8.1). Standard VAE: mean $\mu \approx 0$, mean $\sigma \approx 0.80$, collapse ratio $= 0.705$. Encoder outputs close to prior. Latent code carries little information. Collapse occurs when decoder strong enough to reconstruct without latent code. KL pushes posterior to prior early, before latent variables used.

### 4.4 Mitigating Posterior Collapse

**$\beta$-annealing:** start $\beta = 0$, increase to 1. Decoder learns to use latent before regularization. Results: collapse ratio $=$ $0.505$ (vs 0.705). Reconstructions sharper, generations more diverse. Gain $= 0.201$. Cyclical $\beta$ tested, improvement similar.

**Summary Table**

| Model | Collapse Ratio | Recon Quality | Generation Diversity |
|---|---|---|---|
| Standard VAE | 0.705 | Good, blurry | Moderate |
| $\beta$-annealed VAE | 0.505 | Good, sharper | Better |
| Cyclical $\beta$ VAE | $\sim$0.5 | Similar | Similar |

**Visual Evidence**

Reconstructions close to originals. Generations diverse but blurred. $\beta$-annealing improves both.

# Task 5

### 5.1 Zero-Shot Classification on STL-10

Plain labels accuracy $= 0.9389$. Simple prompts accuracy $= 0.9485$. Descriptive prompts accuracy $= 0.9483$. Prompting with text improves zero-shot accuracy. Simple vs descriptive difference negligible. Both better than plain labels.

### 5.2 Exploring the Modality Gap

t-SNE and UMAP show distinct image and text clusters before alignment. Clear modality gap. After normalization, mean norm $\approx 1.0$ for both modalities. Gap remains in similarity scores. Diagonal similarity (correct pairs) $= 0.2791$. Off-diagonal $= 0.2175$. CLIP still high accuracy $\sim 0.95$. Relative similarity preserved, correct ¿ incorrect.

### 5.3 Bridging the Modality Gap

Procrustes alignment reduces gap. Plots show image and text embeddings closer, overlapping. Similarity scores: before $= 0.2791$, after $= 0.8903$, gain $= +0.6113$. Classification: original $= 0.9500$, aligned $= 1.0000$, gain $= +0.0500$. Alignment boosts similarity and accuracy to perfect on subset.

**Visual Evidence**

Before alignment: clusters separated (blue vs red). After alignment: overlap. Gap reduced.

# 5. Conclusion

# References