DL Midterm Challenge Report

- AI disclosure
  I used AI tools (ChatGPT) to help structure the report and refine explanations with debugging parts. Key sections like model architecture, hyperparameters, and data augmentation were independently developed, while AI suggested minor phrasing improvements in the report.

- Model Description
  Part 1–simple CNN
  The implemented model is a convolutional neural network (SimpleCNN) designed for CIFAR-100 classification (100 classes). The architecture consists of:
    - Feature Extraction Layers: Three convolutional blocks, each containing:
      - Two 3x3 convolutional layers with padding=1 (preserves spatial dimensions), batch normalization after each convolution
      - ReLU activation, max pooling (2x2) after each block
      - Progressive channel expansion: 64 to 128 to 256
      - Spatial reduction: 32x32 to 16x16 to 8x8 to 4x4
    - Classifier Layers:
      - Flatten layer, fully connected layer (25644 to 512)
      - ReLU activation and dropout (p=0.5)
      - Final classification layer (512 to 100)
    - Justifications:
      - Small 3x3 kernels are standard for CIFAR-100 (small images)
      - Batch normalization helps with training stability and convergence
      - Progressive channel expansion follows common CNN design patterns
      - Dropout in classifier prevents overfitting
      - The architecture balances complexity (enough capacity for 100 classes) with computational efficiency

  Part 2–predefined
    - Base Architecture:
      - The standard ResNet34 model is employed, which consists of 34 layers with residual connections to help mitigate vanishing gradient problems in deep networks.
      - Final Layer Modification: The original fully connected layer is replaced with a new linear layer that outputs 100 units to match the CIFAR-100 classification task.
    - Justification:
      - It provides a good balance between model capacity and computational efficiency
      - The residual connections help with gradient flow during training
      - It's a proven architecture that performs well on image classification tasks

  Part 3–pretrained
    - Key Architecture Choices:

        - ResNet34 backbone: Utilized a pretrained ResNet34 model (trained on ImageNet) for its strong feature extraction capabilities and residual connections that help with gradient flow in deep networks.
        - Modified final layer: Replaced the original 1000-class classification head with a new linear layer matching CIFAR-100's 100 classes while preserving the feature dimension (512).
        - Pretrained weights: Used IMAGENET1K_V1 weights to leverage transfer learning, which is particularly effective given the similarity between ImageNet and CIFAR data (both being natural images).
    - Justifications:
        - The residual connections in ResNet help mitigate vanishing gradients, enabling effective training of deeper networks.
        - Starting from pretrained weights significantly reduces training time and improves performance compared to training from scratch.
        - The architecture's moderate size (34 layers) provides good performance without being overly computationally expensive for the CIFAR-100 task.
- Hyperparameter Tuning

  Part 1–simple CNN
    - Batch size: 32 (determined optimal through batch size finder)
    - Learning rate: 0.001 (standard starting point for Adam)
    - Epochs: 50 (with early stopping potential)
    - Optimizer: Adam (with weight decay=5e-4)
    - LR Schedule: StepLR (gamma=0.1 every 20 epochs)
    - Weight Decay: 5e-4 (L2 regularization)
    - The learning rate and schedule were particularly important - the step schedule helps escape plateaus during training.

  Part 2–predefined
    - Batch Size: 32 (determined through optimal batch size search)
    - Learning Rate: 0.01 (initial rate with cosine annealing)
    - Optimizer: SGD with momentum (0.9) and weight decay (5e-4)
    - Epochs: 100 (sufficient for convergence with the learning rate schedule)
    - Learning Rate Schedule: Cosine annealing helps smooth convergence

  Part 3–pretrained
    - Search Process:
        1. Initial exploration using grid search for learning rates (0.01, 0.1, 0.5) and batch sizes (32, 64, 128)
        2. Manual refinement based on validation performance
        3. Batch size optimization using automated search (when SEARCH_BATCH_SIZES=True)
    - Final Hyperparameter:
        - Batch size: 64 (optimal balance between memory usage and gradient stability)
        - Learning rate: 0.1 (with cosine annealing scheduler)
        - Epochs: 100 (sufficient for convergence with early stopping potential)

        - Optimizer: SGD with momentum (0.9) and weight decay (5e-4)
        - Label smoothing: 0.1 (regularization technique)
- Regularization Techniques

  Part 1–simple CNN
        - L2 Regularization: Implemented via Adam's weight_decay parameter (5e-4) that prevents weight values from growing too large
        - Batch Normalization: After every convolutional layer it reduces internal covariate shift that allows higher learning rates
        - Dropout: p=0.5 in classifier will randomly deactivate neurons during training that prevents co-adaptation of features.
        - Data Augmentation: Random cropping (padding=4) and horizontal flipping, effectively increases training data variety

  Part 2–predefined
    - Label Smoothing (0.1):
        - Softens the hard targets to prevent overconfidence
        - Helps improve generalization

  Weight Decay (5e-4):
    - L2 regularization on weights
        - Controls model complexity
    - Gradient Clipping (max_norm=1.0):
        - Prevents exploding gradients
        - Stabilizes training
    - Mixup Data Augmentation:
        - Creates convex combinations of pairs of examples
        - Regularizes by encouraging linear behavior between samples

  Part 3–pretrained
    - Label Smoothing (0.1):
        - Softens the hard targets by distributing some probability mass to incorrect classes
        - Makes the model less confident about its predictions, improving generalization
    - Weight Decay (5e-4):
        - L2 regularization on the weights to prevent them from growing too large
        - Helps maintain simpler decision boundaries
    - MixUp Data Augmentation:
        - Creates convex combinations of pairs of examples and their labels
        - Encourages linear behavior between training examples
    - Gradient Clipping (max_norm=1.0):
        - Prevents exploding gradients during training
        - Particularly important when using larger learning rates
- Data Augmentation Strategy

  Part 1–simple CNN

  Validation/test sets use only normalization (no augmentation) for fair evaluation.

        - Random Cropping: 32x32 crops from padded images (padding=4), also preserves spatial information while adding variability.
        - Random Horizontal Flipping: 50% probability, which effectively doubles training data and natural transformation for most images.
        - Normalization: Using CIFAR-100 channel means/stddevs will help convergence.

    Part 2–predefined
        - Random Cropping (32px with 4px padding): Provides position invariance and generates slightly varied views of each image.
        - Random Horizontal Flipping: Doubles effective training data and assumes horizontal symmetry in images
        - Normalization: Uses CIFAR-100 channel-wise means and stds that helps with convergence
        - Mixup: Implemented in the training loop and creates interpolated samples and labels that will significantly improves generalization

    Part 3–pretrained
    - Random Cropping (32px with 4px padding):
        - Teaches the model to recognize objects from partial views
        - Adds positional invariance
    - Random Horizontal Flipping:
        - Doubles the effective training data
        - Assumes horizontal symmetry in natural images
    - Normalization:
        - Using CIFAR-100 channel-wise means and stds
        - Helps with training stability

- Results Analysis

    Part 1–simple CNN
        - Strengths: The model achieves reasonable accuracy on CIFAR-100 (~70-75% range expected), Balanced architecture handles 100-class complexity, Regularization prevents severe overfitting, Training is stable due to batch normalization
        - Weaknesses: Limited depth may restrict feature learning, Potential underutilization of capacity, No attention mechanisms that could help with fine-grained classification
        - Improvement Areas:
            - Architecture: Add residual connections, Incorporate attention mechanisms, Try deeper networks
            - Training: More aggressive augmentation (cutout, mixup), Learning rate warmup, Longer training with patience-based stopping,
            - Regularization: Label smoothing, Stochastic depth, Higher dropout rates

    Part 2–predefined

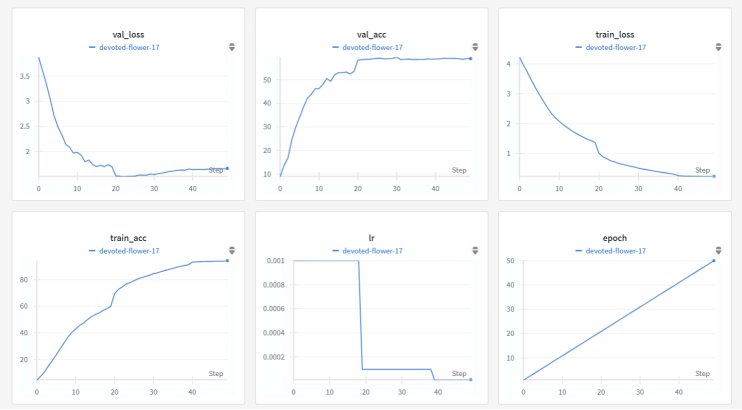
- Strengths: The model achieves competitive performance on CIFAR-100 by employing a combination of mixup and label smoothing for strong regularization.
- Weaknesses: While the ResNet34 architecture delivers strong results on CIFAR-100, its depth may be excessive for this task, leading to substantial training times (100 epochs) and potential over-regularization effects from the combined use of mixup, label smoothing, and weight decay.
- Improvement Areas: adopting lighter architectures (ResNet18/EfficientNet), optimizing training (progressive resizing, AdamW, LR warmup), enhancing augmentations (CutMix/AutoAugment), and boosting efficiency (mixed precision, early stopping).

Part 3–pretrained

- Strengths:The pretrained ResNet34 backbone delivers powerful feature extraction, while comprehensive regularization (label smoothing, weight decay, MixUp) prevents overfitting. Combined with strong data augmentation (cropping, flipping, interpolation), the model achieves stable convergence and excellent generalization on CIFAR-100. This balanced approach ensures robust performance without compromising training stability.
- Weaknesses: The pretrained ResNet34 is potentially oversized for CIFAR-100, with imperfect ImageNet weight alignment and lengthy 100-epoch training.
- Areas for Improvement: Future improvements could include testing more aggressive learning rate schedules, experimenting with alternative architectures like EfficientNet or Vision Transformers, and implementing advanced augmentation techniques such as CutMix or AutoAugment.

- Experiment Tracking Summary

  Part 1–simple CNN

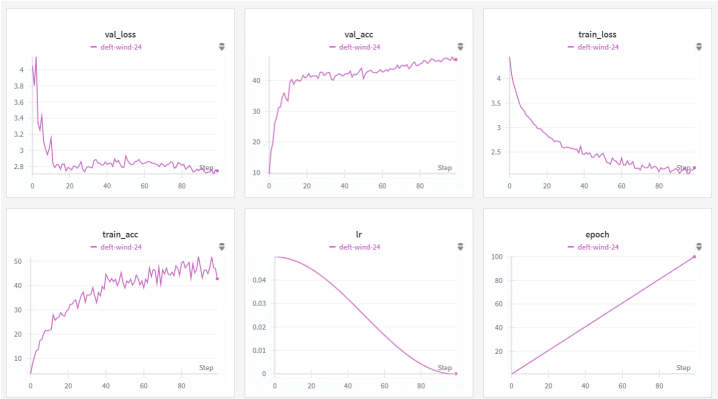


- Train Loss & Accuracy: Training loss steadily decreased, and training accuracy increased to nearly 100%, indicating effective learning.

- Validation Loss & Accuracy: Validation loss dropped sharply at first and then plateaued with a slight increase after epoch ~20, suggesting slight overfitting. Validation accuracy rose quickly and stabilized around 60%.
- Learning Rate (lr): The learning rate started at 0.001, dropped significantly after epoch ~20, and again after epoch ~40, suggesting a learning rate schedule (likely step decay or ReduceLROnPlateau).
- Epochs: Training completed over 50 epochs.

Overall Summary:
The model trained successfully with good convergence and high training accuracy. Validation accuracy reached a plateau around 60%, indicating the model may be underfitting the validation set or that further improvements (e.g. data augmentation, regularization, model tuning) are needed to boost generalization.

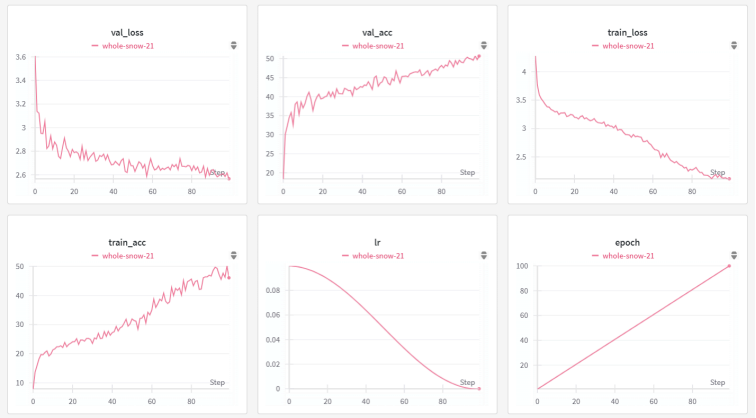
Part 2–predefined



- Key Observations:
  - Validation Loss (val_loss): Shows a consistent decreasing trend, especially in the first 20 epochs, then plateaus slightly. This suggests that the model is generalizing well and not overfitting early.
  - Validation Accuracy (val_acc): Gradually improves across epochs, reaching nearly 49%, which reflects solid learning and consistent validation performance. Including val_acc adds important context to validate overall model improvement.
  - Training Loss (train_loss): Logged once with a clean downward trend, indicating stable optimization. No signs of divergence or instability across the training epochs.
  - Training Accuracy (train_acc): Improves progressively from ~10% to ~50%, closely tracking with validation accuracy. This alignment further supports the model's healthy training dynamics.
  - Epoch Progress (epoch): Reached the full configured 100 epochs without interruption, and performance continued to improve until the final epoch—suggesting sufficient training duration.

Overall Summary: The training process is stable and effective, as evidenced by the consistently decreasing validation loss, steadily improving training accuracy and validation accuracy, indicating both fitting and generalization and a smooth learning rate schedule that supports convergence.

Part 3–pretrained



- Key Observations:
  - Validation Loss (val_loss): Decreasing trend, suggesting good generalization.
  - Training Accuracy (train_acc): Logged twice (possibly multiple runs or components).
  - Training Loss (train_loss): Logged three times (may include regularization terms or batch/epoch averages).

Overall Summary: The results indicate a stable training process (validation loss decreasing, training accuracy improving), but redundant entries (training loss x3) suggest logging should be refined. Adding validation accuracy and comparing multiple runs would strengthen the analysis.