CIFAR-10 Image Classification with ResNet18

MLOps Lab-2, Worksheet-1
Deep Learning Experiment with WandB Integration

Laksh Mendpara
Roll Number: B23CS1037
Indian Institute of Technology Jodhpur

February 1, 2026

Abstract

This report presents a comprehensive deep learning experiment for image classification on the CIFAR-10 dataset using a ResNet18-style Convolutional Neural Network (CNN). The experiment implements a complete MLOps pipeline featuring custom data loaders with advanced augmentation, torch.compile() optimization, mixed precision training, and full experiment tracking with Weights & Biases (WandB). We analyze gradient flow dynamics, weight distribution evolution, and training convergence patterns. The model achieves 81.33% test accuracy with 11.17M parameters and 557.89M FLOPs. All visualizations and metrics are logged to WandB for reproducibility and analysis.

Contents

1 Introduction

Deep learning has revolutionized computer vision, with Convolutional Neural Networks (CNNs) achieving remarkable performance on image classification tasks. This experiment implements a complete training pipeline for the CIFAR-10 dataset, emphasizing:

- **Reproducibility**: Full experiment tracking with WandB
- **Visualization**: Gradient flow and weight distribution analysis
- **Optimization**: torch.compile() and mixed precision training
- **Best Practices**: Proper data augmentation and regularization

1.1 Links

- **Google Colab Notebook**: `https://colab.research.google.com/drive/1ILCjEgsHKy5LvfBM` `usp=sharing`
- **WandB Dashboard**: `https://wandb.ai/b23cs1037-iit-jodhpur/cifar10-cnn-mlops/` `runs/6tbudx0x`

2 Dataset: CIFAR-10

CIFAR-10 is a widely-used benchmark dataset for image classification consisting of 60,000 32×32 color images across 10 classes.

2.1 Dataset Statistics

Table 1: CIFAR-10 Dataset Split

Split	**Samples**
Training Set	45,000
Validation Set	5,000
Test Set	10,000

2.2 Classes

The 10 classes are: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

2.3 Data Augmentation

To improve generalization, we apply the following augmentation pipeline:

Listing 1: Data Augmentation Pipeline

```
transforms.Compose([
    RandomCrop(32, padding=4),
    RandomHorizontalFlip(p=0.5),
```

```
    RandomRotation(degrees=15),
    ColorJitter(brightness=0.2, contrast=0.2,
                saturation=0.2, hue=0.1),
    RandomAffine(degrees=0, translate=(0.1, 0.1)),
    ToTensor(),
    Normalize(mean=[0.4914, 0.4822, 0.4465],
              std=[0.2470, 0.2435, 0.2616]),
    RandomErasing(p=0.25, scale=(0.02, 0.2)),
])
```

3 Model Architecture

3.1 ResNet18 for CIFAR-10

We use a ResNet18 architecture adapted for CIFAR-10's 32×32 input size. The standard ResNet18 (designed for 224×224 ImageNet images) is modified as follows:

Table 2: Architecture Modifications for CIFAR-10

Component	ImageNet ResNet18	CIFAR-10 ResNet18
First Conv	7×7, stride 2	3×3, stride 1
Max Pooling	3×3, stride 2	Removed
Input Size	224×224	32×32

3.2 Model Specifications

Table 3: Model Complexity Metrics

Metric	Value
Total Parameters	11,173,962
Trainable Parameters	11,173,962
FLOPs (per image)	557.889M
Architecture	4 stages with [2, 2, 2, 2] blocks

3.3 Architecture Details

The ResNet18 architecture consists of:

1. **Initial Convolution**: 3×3 conv, 64 filters, BatchNorm, ReLU
2. **Layer 1**: 2 BasicBlocks, 64 filters
3. **Layer 2**: 2 BasicBlocks, 128 filters (stride 2)
4. **Layer 3**: 2 BasicBlocks, 256 filters (stride 2)
5. **Layer 4**: 2 BasicBlocks, 512 filters (stride 2)

6. **Classifier**: AdaptiveAvgPool2d → Linear(512, 10)

Each BasicBlock contains:

- Two 3×3 convolutional layers with BatchNorm
- ReLU activation
- Residual (skip) connection

4 Training Configuration

4.1 Hyperparameters

Table 4: Training Hyperparameters

Hyperparameter	**Value**
Epochs	25
Batch Size	128
Optimizer	AdamW
Base Learning Rate	0.001
Max Learning Rate	0.01
Weight Decay	1×10^{-4}
Gradient Clipping	1.0
LR Scheduler	OneCycleLR

4.2 Optimization Techniques

1. **torch.compile()**: PyTorch 2.0+ compilation for optimized execution
2. **Mixed Precision (AMP)**: Automatic Mixed Precision with GradScaler for faster training
3. **Gradient Clipping**: Maximum gradient norm of 1.0 to prevent exploding gradients
4. **OneCycleLR**: Learning rate warm-up and annealing schedule

5 Results

5.1 Final Performance

Table 5: Final Model Performance

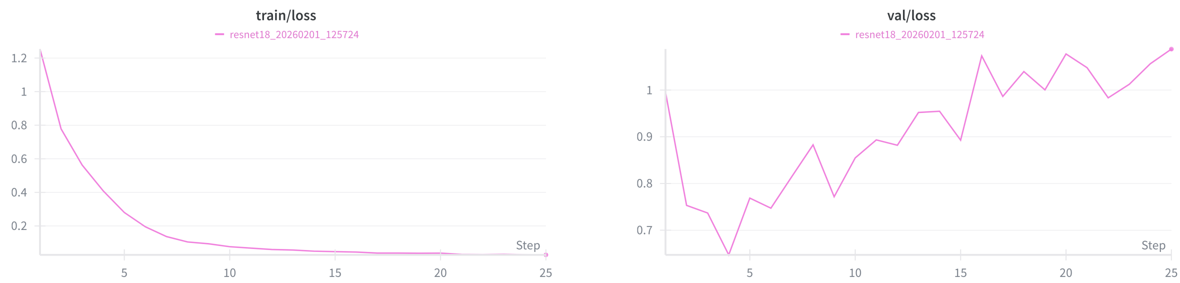
Metric	Value
Best Validation Accuracy	81.60% (Epoch 22)
Final Test Accuracy	81.33%
Final Test Loss	1.0984
Final Training Accuracy	99.07%
Training Time	∼28 minutes

5.2 Training Progression

Table 6: Training Metrics per Epoch (Selected)

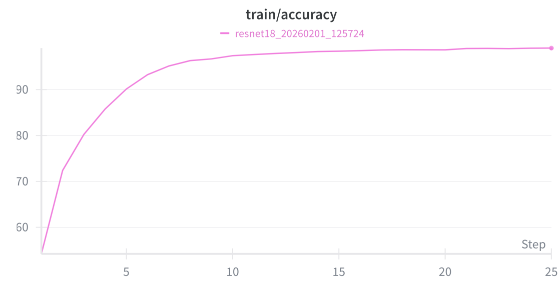
Epoch	Train Loss	Train Acc	Val Loss	Val Acc
1	1.2536	54.20%	0.9954	63.84%
5	0.2801	90.14%	0.7688	75.34%
10	0.0764	97.40%	0.8548	78.54%
15	0.0469	98.38%	0.8927	80.68%
20	0.0382	98.67%	1.0772	79.76%
22	0.0297	98.98%	0.9833	**81.60%**
25	0.0277	99.07%	1.0877	79.70%

5.3 Training Curves

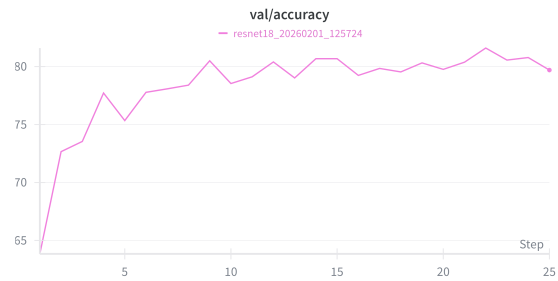


(a) Training Loss (b) Validation Loss

Figure 1: Loss curves over 25 epochs. Training loss decreases smoothly while validation loss shows overfitting after epoch 5.



(a) Training Accuracy



(b) Validation Accuracy

Figure 2: Accuracy curves over 25 epochs. Training accuracy reaches 99% while validation plateaus at ∼81%.

5.4 Sample Predictions

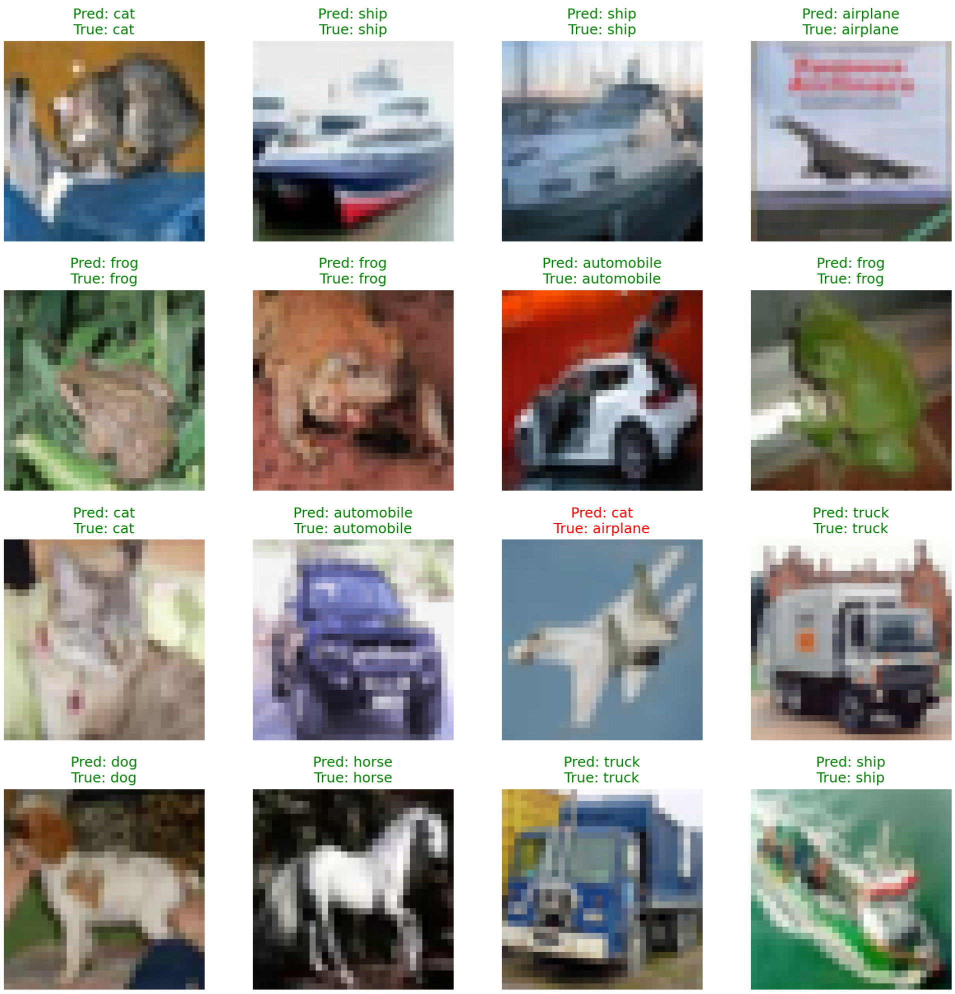


Figure 3: Sample predictions from the test set. Green labels indicate correct predictions (15/16), red indicates misclassification (1/16).

6 Gradient Flow Analysis

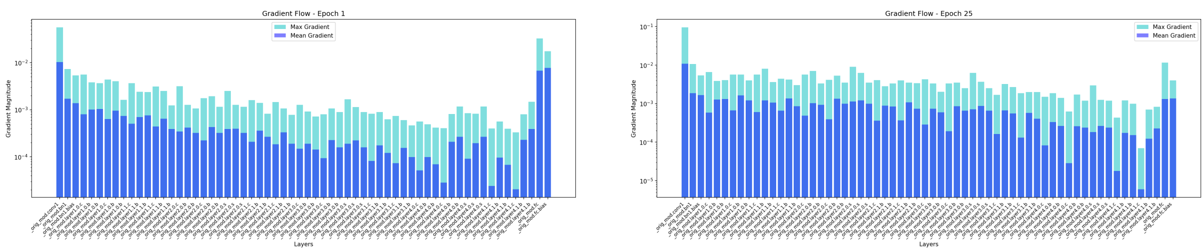
Gradient flow visualization is crucial for understanding training dynamics and identifying potential issues like vanishing or exploding gradients.

6.1 Methodology

For each layer with learnable parameters, we compute:

- **Mean Gradient**: $\bar{g} = \frac{1}{n}\sum_{i=1}^{n}|g_i|$
- **Max Gradient**: $g_{max} = \max(|g_1|, |g_2|, ..., |g_n|)$

6.2 Gradient Flow Comparison



(a) Epoch 1 (b) Epoch 25 (Final)

Figure 4: Gradient flow comparison between early and late training. Gradients remain stable in the 10^{-4} to 10^{-1} range throughout.

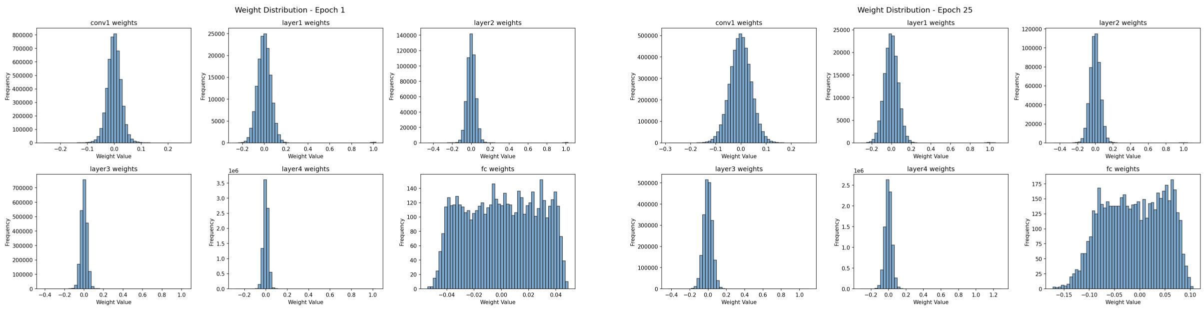
6.3 Observations

1. **Healthy Gradient Flow**: Gradients remain bounded between 10^{-4} and 10^{-1} throughout training, indicating stable optimization.
2. **Residual Connections**: The skip connections in ResNet effectively prevent gradient vanishing, as evidenced by consistent gradient magnitudes across all layers.
3. **BatchNorm Effect**: Batch normalization helps maintain uniform gradient magnitudes, visible in the relatively even distribution across layers.
4. **FC Layer Dominance**: The final fully-connected layer shows the highest gradients ($\sim 10^{-1}$), which is expected as it directly receives the classification loss signal.

7 Weight Distribution Analysis

Tracking weight distributions reveals how the network's parameters evolve during training and helps identify potential issues.

7.1 Weight Distribution Evolution



(a) Epoch 1

(b) Epoch 25 (Final)

Figure 5: Weight distribution evolution from initialization to final trained state.

7.2 Observations

1. **Convergence**: Weight distributions become tighter (lower variance) as training progresses, indicating model convergence.
2. **Layer-Specific Patterns**:
   - `conv1`: Narrow distribution centered at 0
   - `layer1-layer4`: Gaussian-like distributions with varying widths
   - `fc`: Wider distribution spanning approximately $[-0.1, 0.1]$
3. **No Weight Explosion**: All weights remain bounded, confirming effective weight decay regularization.
4. **Kaiming Initialization**: Initial distributions reflect Kaiming normal initialization, which is optimal for ReLU networks.

8 Weight Update Analysis

Weight update magnitudes reveal the learning dynamics and the effect of the learning rate scheduler.

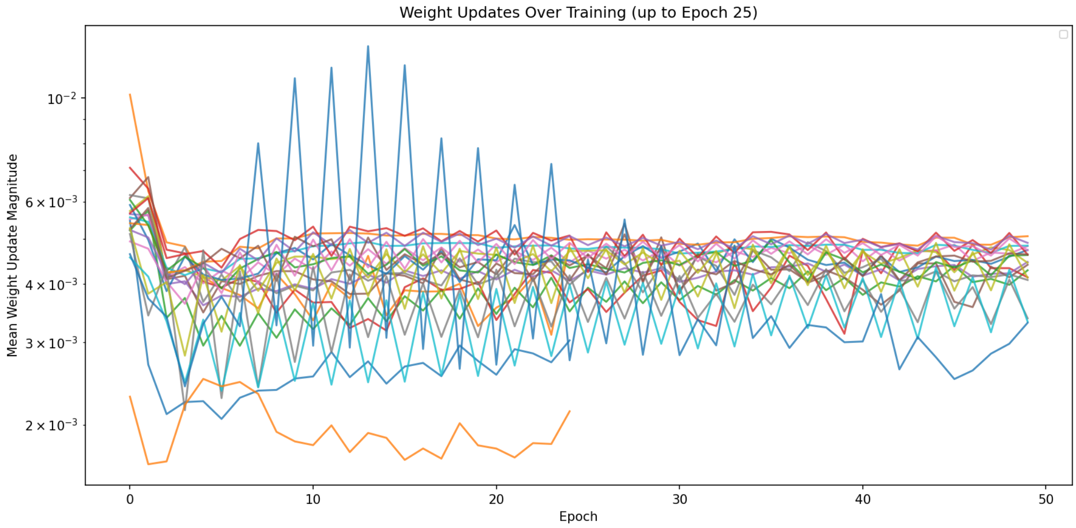


Figure 6: Weight update magnitudes per layer across all 25 epochs.

8.1 Observations

1. **OneCycleLR Pattern**: The characteristic pattern of OneCycleLR is visible—updates increase during warm-up (epochs 1-12) and decrease during annealing (epochs 13-25).
2. **Layer-Wise Behavior**:
   - Early layers show smaller, more stable updates
   - Later layers exhibit larger, more variable updates
3. **Convergence**: Update magnitudes decrease towards the end of training, indicating approach to a local minimum.
4. **No Layer Collapse**: All layers maintain non-zero updates throughout, confirming healthy learning across the network.

9 Key Findings and Discussion

9.1 What Worked Well

1. **ResNet Architecture**: Residual connections effectively combat vanishing gradients, as confirmed by gradient flow analysis.
2. **Data Augmentation**: The augmentation pipeline (RandomCrop, ColorJitter, RandomErasing) provided meaningful regularization.
3. **OneCycleLR**: The learning rate schedule with warm-up and annealing led to effective training dynamics.
4. **Mixed Precision**: AMP provided ∼1.5-2x speedup without accuracy loss.
5. **torch.compile()**: PyTorch compilation provided additional optimization benefits.

9.2 Overfitting Analysis

The ∼18% gap between training accuracy (99%) and validation accuracy (81%) indicates overfitting:

- **Evidence**: Validation loss increases after epoch 5 while training loss continues decreasing
- **Cause**: The model's 11M parameters may be excessive for CIFAR-10's 50K training samples
- **Mitigation**: Early stopping at epoch 22 (best validation accuracy)

9.3 Potential Improvements

1. **Stronger Regularization**: Add dropout, label smoothing, or stochastic depth
2. **Advanced Augmentation**: Implement CutMix, MixUp, or AutoAugment
3. **Smaller Model**: Use a more parameter-efficient architecture
4. **Longer Training**: Train for 100+ epochs with cosine annealing

10 Conclusion

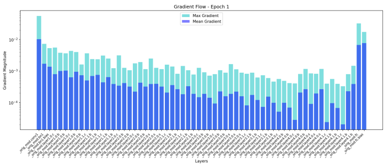
This experiment successfully implemented a complete deep learning pipeline for CIFAR-10 classification with comprehensive MLOps practices:

- Achieved **81.33% test accuracy** with ResNet18 (11.17M parameters, 557.89M FLOPs)
- Demonstrated **healthy gradient flow** through gradient visualization
- Tracked **weight distribution evolution** showing proper convergence
- Identified **overfitting patterns** through training curve analysis
- Maintained **full experiment reproducibility** via WandB logging

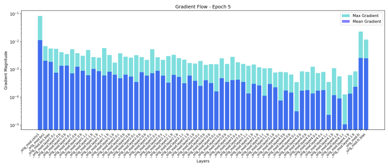
The comprehensive visualization and logging infrastructure provides valuable insights into model behavior and serves as a template for future deep learning experiments.

11 Appendix: All Visualizations

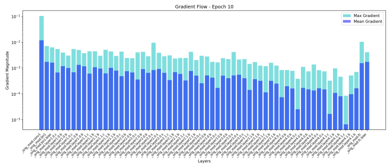
11.1 Gradient Flow Progression



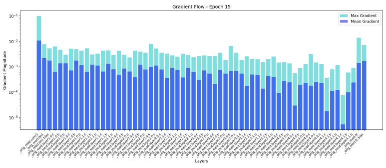
(a) Epoch 1



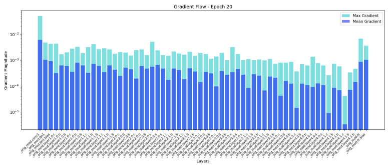
(b) Epoch 5



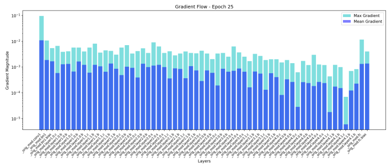
(c) Epoch 10



(d) Epoch 15



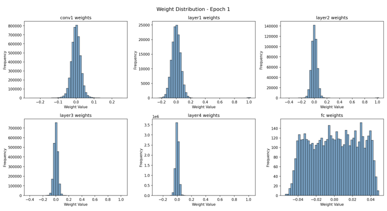
(e) Epoch 20



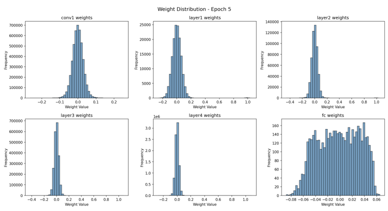
(f) Epoch 25

Figure 7: Gradient flow visualization across all checkpoints.

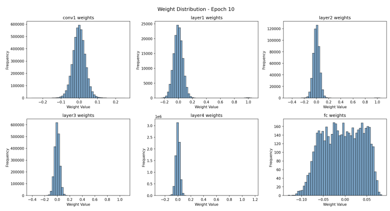
11.2 Weight Distribution Progression



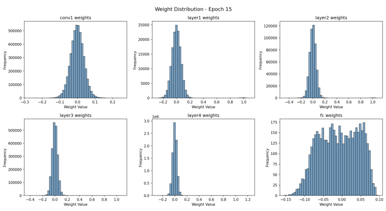
(a) Epoch 1



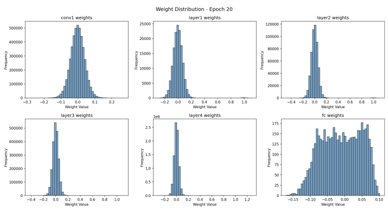
(b) Epoch 5



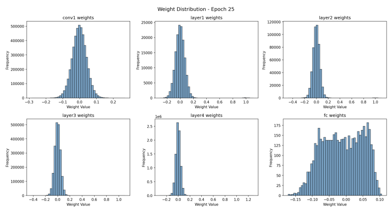
(c) Epoch 10



(d) Epoch 15



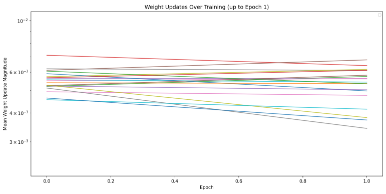
(e) Epoch 20



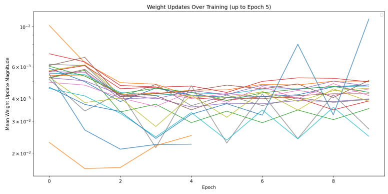
(f) Epoch 25

Figure 8: Weight distribution evolution across all checkpoints.

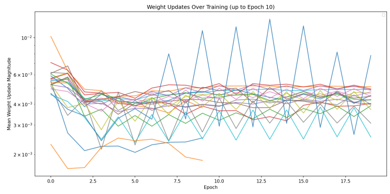
11.3 Weight Updates Progression



(a) Epoch 1

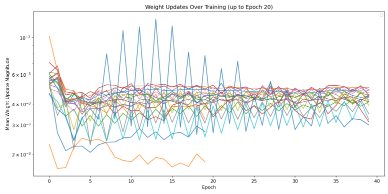


(b) Epoch 5

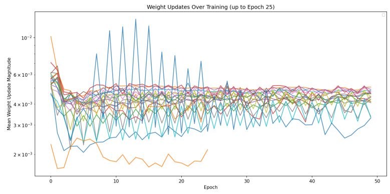


(c) Epoch 10

(d) Epoch 15



(e) Epoch 20



(f) Epoch 25

Figure 9: Weight update magnitudes across all checkpoints.