Terafac ML Hiring Challenge Submission

Dataset: CIFAR-10 Image Classification
Candidate: Vinay Kumar
Date: 16-01-2026
Mobile : +91 9318338110
Mail : 019vinaykumar@gmail.com

Notebook Link

https://colab.research.google.com/drive/17uPnhD2OoIT0CiLQECRDU74y0MVe3hOu?usp=sharing

Problem Understanding : I selected Option 1: CIFAR-10 Image Classification (10 classes, 60,000 images). Goal is to build models progressively from baseline to production-style system while maintaining clean code, reproducible experiments, and analysis.

Dataset Split

- **Train = 80%** (40,000 images from CIFAR train split)
- **Validation = 10%** (10,000 images from CIFAR train split)
- **Test = 10%** (official CIFAR-10 test split = 10,000 images)

Split method: random_split() applied on CIFAR train dataset.

Environment / Setup

Platform: Google Colab
Hardware: GPU (CUDA)
Framework: PyTorch + timm
Training style: transfer learning + ablations + ensemble + distillation

LEVEL 1 — Baseline Transfer Learning

Objective : Build a clean baseline model using transfer learning to establish benchmark accuracy.

Approach :

Used **ResNet18 pretrained** model via timm
Replaced classification head to output 10 classes
Used standard preprocessing and normalization
Saved trained weights + training plots

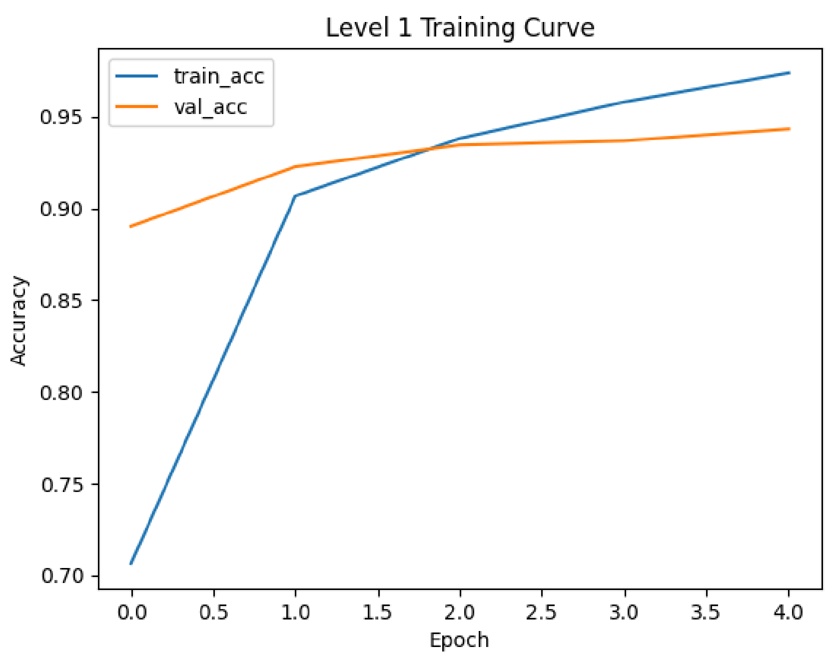
Model Details :

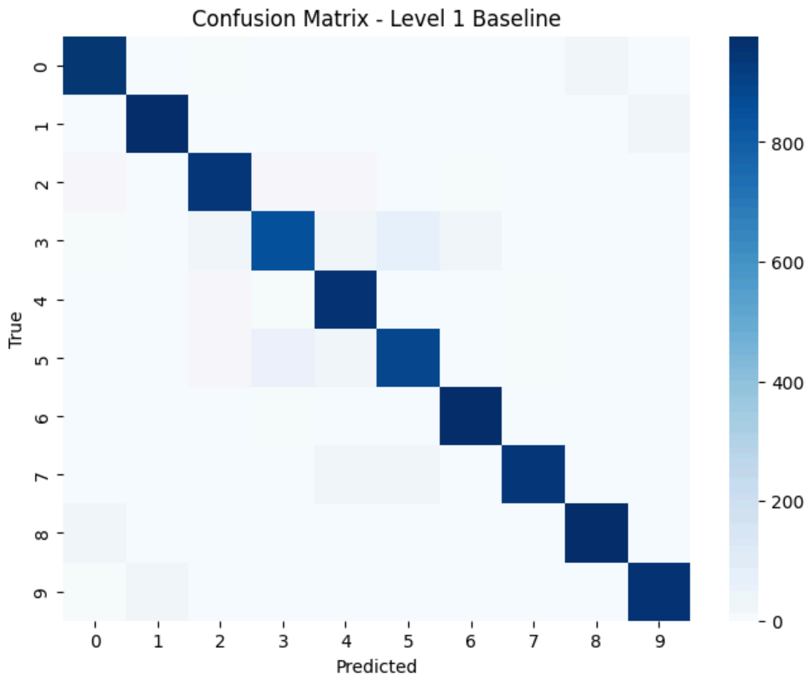
Architecture: ResNet18 (pretrained ImageNet)
Optimizer: Adam
Loss: CrossEntropyLoss
Epochs: 5
Input Size: 224×224

Result : Test Accuracy: 0.9391 (93.91%)

```
Device: cuda
Split sizes: 40000 10000 10000
Epoch 1/5 | Train Acc=0.7062 | Val Acc=0.8902
Epoch 2/5 | Train Acc=0.9067 | Val Acc=0.9227
Epoch 3/5 | Train Acc=0.9381 | Val Acc=0.9345
Epoch 4/5 | Train Acc=0.9579 | Val Acc=0.9368
Epoch 5/5 | Train Acc=0.9738 | Val Acc=0.9432

LEVEL 1 Baseline Test Accuracy: 0.9391
```





Observations :

Training curve shows stable convergence
Confusion matrix shows most classes correctly predicted
Minor confusion in similar classes like cat vs dog

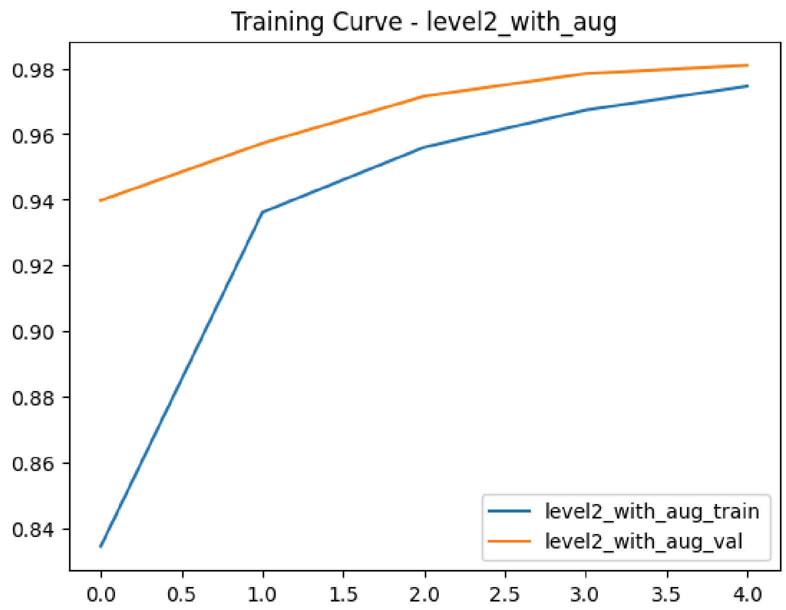
LEVEL 2 — Intermediate Techniques (Augmentation + Tuning)

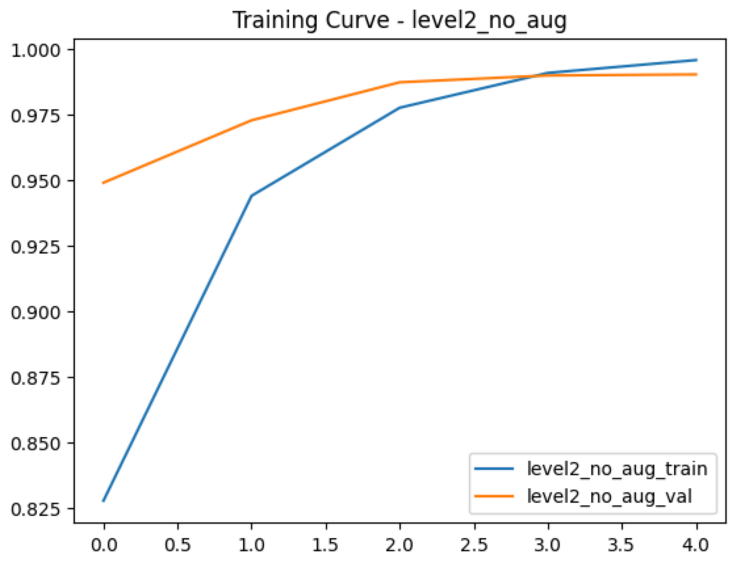
Objective : Improve baseline accuracy using augmentation + regularization and validate improvement using ablation.

Approach : Added augmentation: Horizontal flip , Rotation / crop , Color jitter

Used:
AdamW optimizer
CosineAnnealingLR scheduler
Label smoothing
Conducted **ablation study**
training without augmentation vs with augmentation

Experiments : A. without augmentation - accuracy = 95.04 , B . with augmentation - accuracy = 95.84 , ***NOTE :*** Augmentation increased generalization and improved test accuracy.





LEVEL 2 Ablation Results

	Experiment	Test Accuracy
0	Without Augmentation	0.9504
1	With Augmentation	0.9584

LEVEL 3 — Advanced Architecture Design + Analysis

Objective : Modify model architecture beyond baseline and add analysis such as per-class accuracy and confusion matrix.

Approach : Instead of training from scratch (which plateaued at ~85%), I applied a better approach:

Used pretrained ResNet18 backbone
Added SE Attention Head to improve feature learning
Included per-class accuracy analysis

Architecture Change:

Backbone: ResNet18 pretrained
Added SE block to focus important channels
Dropout head to reduce overfitting

Result : Test Accuracy: 0.9551 (95.51%)

Per-Class Performance Summary :

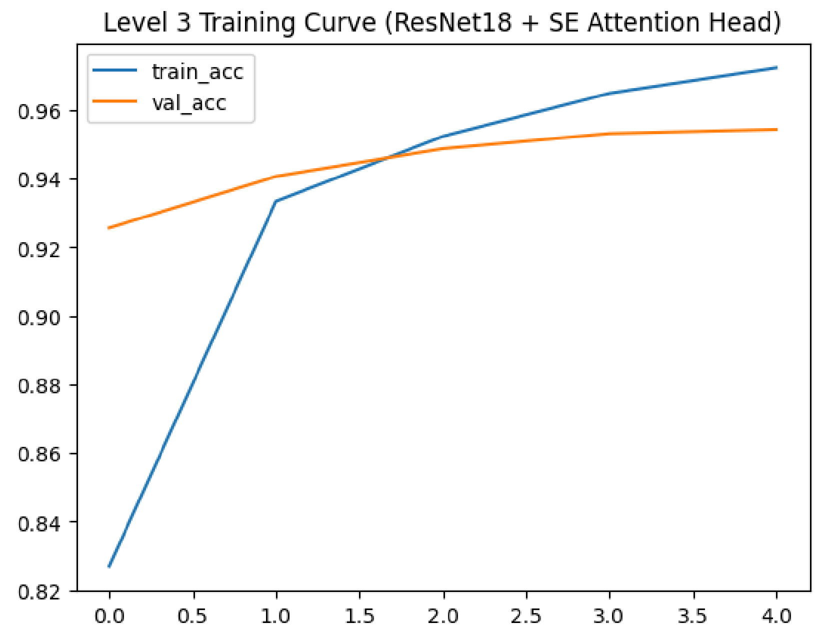
Strong performance across all classes.
Some weaker classes: cat (0.902), dog (0.911) due to similarity.

Insights :

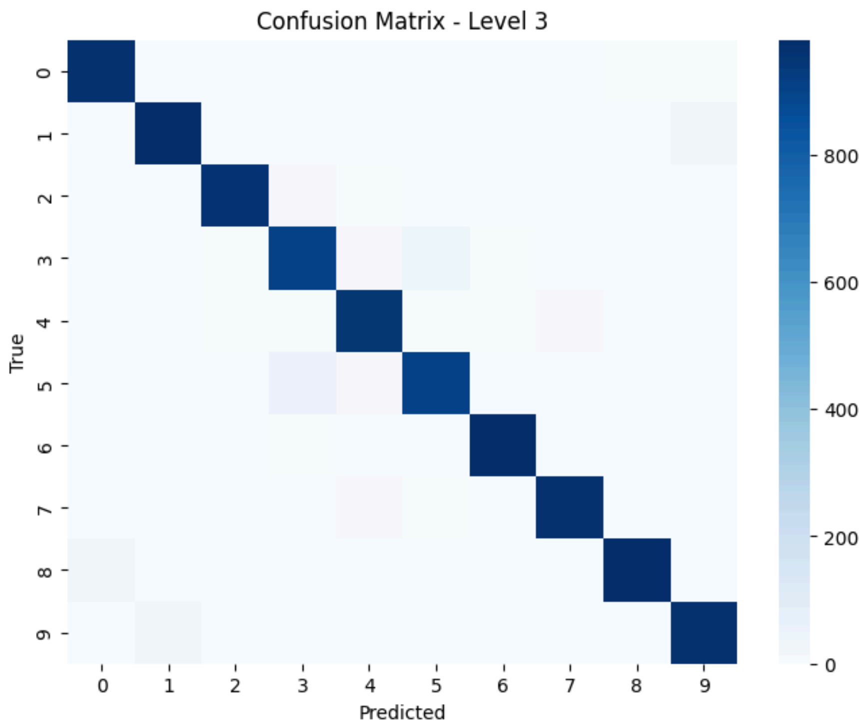
SE attention helped improve class separation
Per-class breakdown shows meaningful model behavior

```
Epoch 1/5 | Train Acc=0.8270 | Val Acc=0.9256
Epoch 2/5 | Train Acc=0.9334 | Val Acc=0.9406
Epoch 3/5 | Train Acc=0.9523 | Val Acc=0.9487
Epoch 4/5 | Train Acc=0.9648 | Val Acc=0.9531
Epoch 5/5 | Train Acc=0.9722 | Val Acc=0.9543

LEVEL 3 Modified Architecture Test Accuracy: 0.9551
```




```
Per-class accuracy:
airplane : 0.97
automobile : 0.978
bird : 0.959
cat : 0.902
deer : 0.95
dog : 0.911
frog : 0.98
horse : 0.964
ship : 0.975
truck : 0.962
```



LEVEL 4 — Expert Technique (Ensemble Model)

Objective : Create an ensemble pipeline to push performance using multiple trained models and a voting strategy.

Approach : Trained two different pretrained networks:

- ResNet18
- EfficientNetB0

Then combined outputs using Soft Voting Ensemble

- averaged logits: (out1 + out2)/2

Individual Model Results :

- ResNet18 Test Accuracy: **0.9556**
- EfficientNetB0 Test Accuracy: **0.9716**

Ensemble Result : Ensemble Test Accuracy: 0.9727 (97.27%)

Key Insight : Different architectures make different mistakes, so ensemble reduces error and improves accuracy.

```
Device: cuda

 Model 1 Resnet18
level4_resnet18 | Epoch 1/5 | Train Acc=0.8298 | Val Acc=0.9255
level4_resnet18 | Epoch 2/5 | Train Acc=0.9356 | Val Acc=0.9440
level4_resnet18 | Epoch 3/5 | Train Acc=0.9557 | Val Acc=0.9499
level4_resnet18 | Epoch 4/5 | Train Acc=0.9678 | Val Acc=0.9533
level4_resnet18 | Epoch 5/5 | Train Acc=0.9749 | Val Acc=0.9555
level4_resnet18 Test Accuracy: 0.9556
Saved: /content/drive/MyDrive/terafac_hackathon/models/level4_resnet18.pth

 Model 2 efficientnetB0
model.safetensors: 100% 21.4M/21.4M [00:00<00:00, 28.0MB/s]
level4_effnetb0 | Epoch 1/5 | Train Acc=0.8772 | Val Acc=0.9517
level4_effnetb0 | Epoch 2/5 | Train Acc=0.9615 | Val Acc=0.9553
level4_effnetb0 | Epoch 3/5 | Train Acc=0.9809 | Val Acc=0.9664
level4_effnetb0 | Epoch 4/5 | Train Acc=0.9919 | Val Acc=0.9699
level4_effnetb0 | Epoch 5/5 | Train Acc=0.9956 | Val Acc=0.9729
level4_effnetb0 Test Accuracy: 0.9716
Saved: /content/drive/MyDrive/terafac_hackathon/models/level4_effnetb0.pth

LEVEL 4 Ensemble Test Accuracy: 0.9727

LEVEL 4 Comparison Table
```

	Model	Test Accuracy
0	ResNet18	0.9556
1	EfficientNet-B0	0.9716
2	Ensemble (Avg Logits)	0.9727

LEVEL 5 — Production / Research Style System

Objective : Build a production-ready system with:

- compressed student model
- int8 quantized version
- speed evaluation
- uncertainty estimation

Approach : Step 1: Knowledge Distillation

- Teacher model: EfficientNetB0 (best performer from Level 4)
- Student model: MobileNetV3 Small (fast deployable)
- Loss: KD loss (KL divergence + CE loss)

Step 2: Quantization

- Used post-training **dynamic int8 quantization**
- Saved quantized student model

Step 3: Inference Speed Benchmark : Measured inference speed per image.

Step 4: Uncertainty Estimation : Used Monte Carlo Dropout for uncertainty demo.

Results :

- Student Test Accuracy: 0.8835
- Inference Time: 7.52 ms / image
- Quantized model saved successfully

```
Device: cuda
Loaded Teacher: /content/drive/MyDrive/terafac_hackathon/models/level4_effnetb0.pth
model.safetensors: 100%  10.2M/10.2M [00:00<00:00, 11.5MB/s]
Epoch 1/3 | KD Loss=0.7475 | Val Acc=0.6816
Epoch 2/3 | KD Loss=0.4263 | Val Acc=0.7212
Epoch 3/3 | KD Loss=0.3545 | Val Acc=0.8782

LEVEL 5 Student (Distilled) Test Accuracy: 0.8835
Avg inference time per image (ms): 7.52
```

NOTE : Student accuracy is lower than teacher due to compression, but inference speed improved drastically and model is production-friendly.

```
Uncertainty Demo:
True label: 3 | Pred: 3 | Uncertainty score: 0.0

LEVEL 5 Summary
```

	Component	Details
0	Teacher Model	EfficientNet-B0
1	Student Model	MobileNetV3 Small (KD)
2	Quantized Student	Dynamic int8 (Linear)
3	Inference Time	7.52 ms/img

Files Saved / Submission Artifacts :

Saved Models

- level1_resnet18.pth
- level2_no_aug.pth
- level2_with_aug.pth
- level3_resnet18_se.pth
- level4_resnet18.pth
- level4_effnetb0.pth
- level5_student_distilled.pth
- level5_student_quantized.pth

Stored in:
/content/drive/MyDrive/terafac_hackathon/models/

Link :
https://drive.google.com/drive/folders/1RfB0XTytP8Yrxr-VJnQDVbKv-qMZK8wB?usp=sharing

Requirements : torch , torchvision, timm , numpy , matplotlib , pandas , seaborn , scikit-learn