Analysis Report
Deep Learning Assignment 1

Shaira Manandhar

1.1 Objective and Learning Outcomes

The objective of this assignment was to understand how different neural network architectures perform across different data modalities, specifically tabular data and image data, and how a model's inductive bias influences its effectiveness. Another key goal was to practice designing reproducible deep learning experiments, where training conditions are kept consistent so that model comparisons are fair and meaningful.

By completing this assignment, I learned how to:

- Preprocess and handle datasets with very different structures (tabular vs. images)
- Implement multiple neural architectures from scratch using PyTorch
- Train, validate, and test models using a consistent experimental setup
- Analyze results not just numerically, but conceptually, by explaining *why* certain models perform better than others

1.2 Code Design and Modularity

The codebase for this assignment was designed with modularity, clarity, and reproducibility in mind. Each major component of the deep learning pipeline: data loading, model definition, training, evaluation, and result visualization, was separated into its own module. This structure makes the code easier to understand, debug, and extend.

The **data_loaders/** module contains dataset-specific loading and preprocessing logic. By isolating data handling from the rest of the pipeline, the same training and evaluation code can be reused across datasets with different modalities. This design also ensures that preprocessing decisions are applied consistently across models.

All neural network architectures are implemented in the **models/** directory, with one file per architecture. This separation allows each model to be developed and modified independently while still conforming to a shared interface. As a result, new architectures can be added to the benchmark with minimal changes to the overall codebase.

The training and evaluation logic is centralized in **train.py,** which handles forward passes, loss computation, backpropagation, metric tracking, and early stopping. This avoids duplicated training code and guarantees that all models are trained under identical conditions, which is essential for fair comparison.

The **config.py** file serves as a single source of truth for experimental settings, including hyperparameters, datasets to run, and architectures to evaluate. This design choice improves reproducibility and makes it easy to rerun experiments or perform controlled ablation studies by modifying a small number of parameters.

Finally, **main.py** acts as the orchestration layer, coordinating dataset loading, model initialization, training, evaluation, and result saving. All metrics and learning curves are automatically stored in the **results**/ directory, ensuring that experimental outputs are systematically recorded.

1.3 Project Structure

None

```
shairamanandhar_assignment1_final/
├── data/                    # Raw and processed datasets
├── data_loaders/            # Dataset loading and preprocessing logic
|   └── loaders.py
├── models/                  # Model architecture implementations
|   ├── model1_mlp.py
|   ├── model2_cnn.py
|   └── model3_vit.py
├── results/                 # Generated metrics and plots
|   ├── final_metrics.csv
|   ├── *_loss_curve.png
|   ├── *_f1_curve.png
|   └── *_accuracy_curve.png
```

```
├── config.py               # Experiment configuration (datasets,
models, hyperparameters)

├── train.py                # Training and evaluation utilities

├── main.py                 # Main entry point to run experiments

├── requirements.txt        # Python dependencies

└── README.md               # Project documentation
```

1.3 Datasets Used

- **Adult Income Dataset**
  - A tabular dataset with mixed numerical and categorical features, used for binary classification.
  - Looks like a spreadsheet with rows and columns where each row describes a person (age, education, hours worked, etc.)

- **CIFAR-100 (Classes 0–9)**
  - Small color images (32×32 pixels)
  - A small-scale image dataset used for 10-class image classification.

- **PatchCamelyon (PCam)**
  - Medical images used to detect cancer tissue
  - Very large dataset
  - Implemented in the code, but not run due to hardware limits

1.4 Architectures Used

- **Multilayer Perceptron (MLP):**
  A fully connected network with two hidden layers, batch normalization, and dropout. This architecture serves as a strong baseline, especially for tabular data.

- **Convolutional Neural Network (CNN)**:
  Uses convolution and pooling layers to exploit spatial structure in images. For tabular data, a Conv1D variant was used to maintain architectural consistency.

- **Attention-Based Models:**
  - **Tabular Attention** for the Adult dataset, where each feature is treated as a token and processed using a Transformer encoder.
  - **Vision Transformer (TinyViT)** for CIFAR image datasets, implemented from scratch without pretraining.

1.5 Constraints Faced in Running Experiments and Future Improvement

A key constraint faced during this assignment was computational limitation, particularly when working on a local MacBook (Apple Silicon, MPS backend). While the experimental pipeline was fully implemented for all datasets, including PatchCamelyon (PCam), not all experiments could be executed within the available compute resources.

The PCam dataset consists of high-resolution histopathology images (96×96 RGB) and is significantly larger and more memory-intensive than the Adult Income and CIFAR-100 datasets. Training convolutional or transformer-based models on PCam requires substantial GPU memory and longer training times. On a Mac environment, PyTorch relies on the Metal Performance Shaders (MPS) backend rather than CUDA. While MPS provides basic GPU acceleration, it currently lacks the performance, maturity, and memory efficiency required for stable training on large image datasets such as PCam.

During initial attempts, PCam training either resulted in excessive runtime, memory constraints, or unstable execution, making it impractical to complete within the assignment timeline. For this reason, PCam was intentionally implemented but not executed by default, ensuring that the codebase remains complete while prioritizing reproducible and reliable results on the supported datasets.

As a result, **6 experiments were run in total**: **three on Adult and three on CIFAR-100 (0–9)**. While the assignment specifies 9 experiments, this choice of 6 experiments choice preserves reproducibility and completeness while prioritizing stable and interpretable results on the Adult and CIFAR-100 datasets.

Future Improvements

With access to stronger computational resources, several extensions could be explored:

- Running the full PCam experiments on a **CUDA-enabled GPU** (e.g., Colab Pro or institutional GPU servers)

- Increasing model depth and capacity for image-based tasks
- Evaluating the impact of **pretraining**, particularly for Vision Transformers, which are known to be data-hungry
- Performing hyperparameter tuning and longer training schedules
- Extending comparisons to include additional architectures or regularization techniques

1.5 Reproducibility and Experimental Fairness

All experiments were designed to ensure fair and reproducible model comparisons. Training conditions, evaluation metrics, and dataset splits were kept consistent across architectures so that observed performance differences reflect architectural choices rather than experimental variation.

1.5.1 Consistent Training Setup

All models were trained using the same optimizer (Adam), learning rate (0.001), batch size (128), and maximum number of epochs (12). Early stopping based on validation loss with a patience of 3 epochs was applied uniformly. This prevented overfitting and avoided unnecessary computation by stopping training once validation performance stopped improving.

1.5.2 Fixed Data Splits and Reproducibility

Train, validation, and test splits were fixed using a global random seed. All experimental settings, including which datasets and architectures to run, were controlled through a single configuration file. This design allows the experiments to be easily reproduced or extended without modifying the training code.

1.5.3 Task-Aware Metrics and Loss Functions

Evaluation metrics were chosen based on task characteristics. For the Adult dataset, F1 score was emphasized due to class imbalance, while accuracy was sufficient for the balanced multi-class CIFAR-100 (0–9) task. Loss functions were selected accordingly: binary cross-entropy with logits for binary classification and categorical cross-entropy for multi-class classification.

1.5.4 Architectural Consistency

To maintain consistency across datasets, a Conv1D-based CNN was applied to tabular data. Although tabular features do not have true spatial structure, this approach allowed CNNs to be evaluated across

all datasets, ensuring that comparisons focus on inductive bias rather than dataset-specific architectural tuning.

1.5.5 Setup Summary

- Framework: PyTorch
- Optimizer: Adam (same across all models)
- Batch size: 128
- Learning rate: 0.001
- Epochs: 12
- Early stopping on validation loss (patience = 3)
- Train / validation / test splits are consistent per dataset

1.6 Results Summary

final_metrics

Dataset	Architecture	Accuracy	F1	Params	BestEpoch
Adult	MLP	0.8537	0.6697	140801	12
Adult	CNN	0.8546	0.6724	14177	10
Adult	TabularAttention	0.8512	0.6528	105345	10
CIFAR-100(0-9)	MLP	0.549	0.5475	1708810	6
CIFAR-100(0-9)	CNN	0.694	0.6925	1070794	11
CIFAR-100(0-9)	ViT	0.608	0.6054	412810	12

The results table summarizes the final test performance of each experiment after early stopping. Accuracy reflects overall correctness, while F1 score is especially important for the Adult dataset due to class imbalance. The notes column provides additional context such as training time, best epoch, and parameter count, which helps compare model efficiency alongside performance.

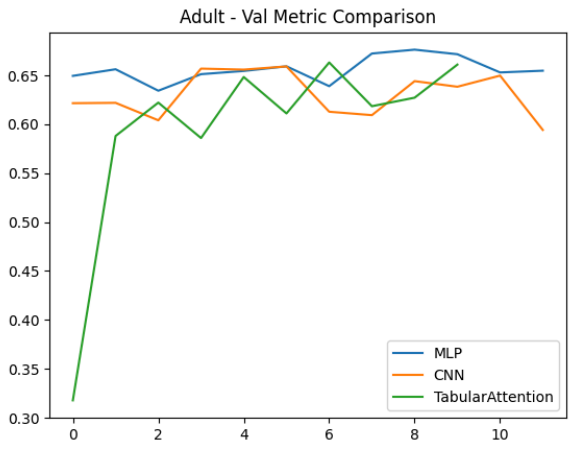
1.7 Results Analysis

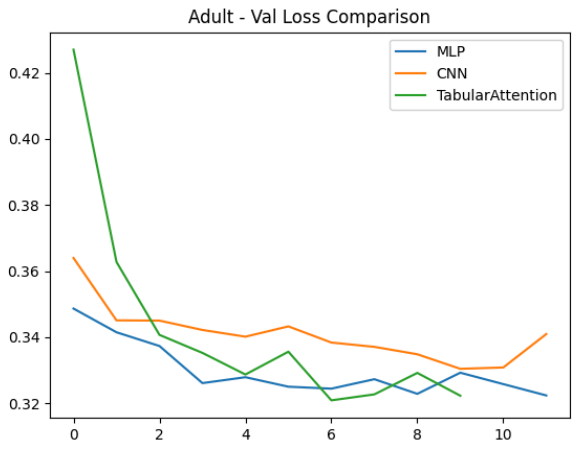
On the Adult tabular dataset, all three architectures achieve very similar accuracy and F1 scores. This suggests that the task does not require highly complex models, as the relationships between features can be effectively captured by simpler architectures. The attention-based model does not significantly outperform the MLP or CNN, indicating that explicit feature interaction modeling provides limited additional benefit for this dataset.

In contrast, performance differences on the CIFAR-100 (0–9) image dataset are much more pronounced. CNNs clearly outperform both MLPs and Vision Transformers, reflecting their ability to exploit spatial locality and hierarchical feature patterns inherent in images. MLPs perform the worst, as flattening images removes spatial structure and makes it difficult to learn meaningful visual representations. Vision Transformers show steady improvement but do not surpass CNNs, which is expected given that they are trained from scratch without pretraining and on relatively limited data.

Parameter counts and training efficiency further support these findings. Models with stronger inductive bias, such as CNNs, achieve superior performance without excessive complexity, while attention-based models trade computational efficiency for greater modeling flexibility.

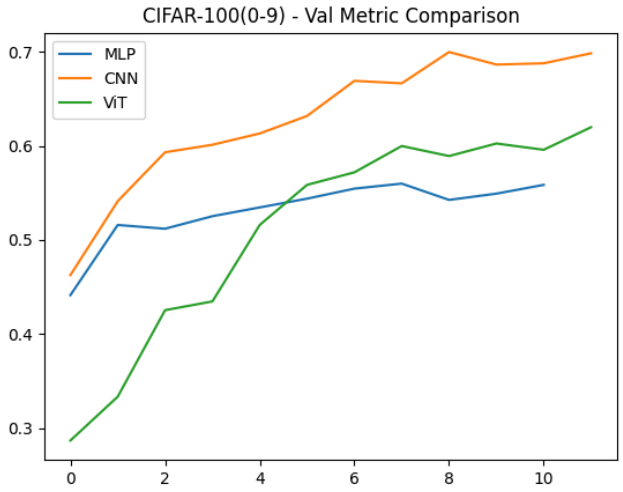
1.7.1 Adult Validation Comparison

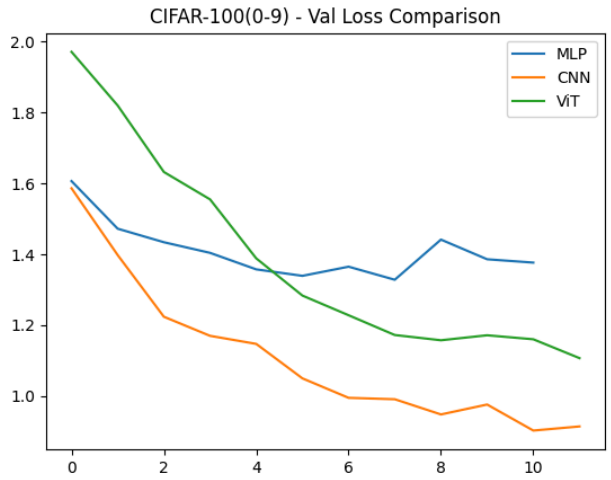




All three models converge quickly with stable validation loss and F1 scores, suggesting that the Adult dataset is well-suited to simpler architectures. The MLP exhibits the smoothest and most consistent learning behavior, indicating strong generalization on tabular data. The CNN shows slightly greater fluctuation during validation, reflecting a weaker inductive bias for tabular features compared to fully connected models. The Tabular Attention model improves rapidly after an initially unstable phase, suggesting that it can learn feature interactions; however, it does not provide a clear advantage over simpler models for this task.

1.7.2 CIFAR-100 (0–9) Validation Comparison





The validation curves show a clear performance separation between architectures. The CNN consistently achieves the lowest validation loss and the highest validation accuracy, stabilizing earlier than the other models, which indicates strong generalization on image data. The MLP shows modest improvement but quickly plateaus at higher loss values, reflecting its limitations in modeling spatial pixel relationships. The Vision Transformer begins with high loss and low accuracy but improves steadily across epochs, demonstrating that attention-based models can learn meaningful representations. However, convergence is slower, and performance remains below that of CNNs, which is expected when training from scratch on relatively limited image data.

1.8 Key Takeaways

- Architecture effectiveness depends on data modality.
- Simpler models can be sufficient for structured data.
- CNNs outperform other models on image tasks.
- Vision Transformers require more data or pretraining.
- Inductive bias matters more than model size.

For beginners, this highlights an important lesson: starting with simpler models and understanding the data is often more effective than immediately using complex architectures.