**Deep Learning Assignment Report – 2024-25**

**Name:** Iason-Christoforos Asproudis  
**Student ID:** p3352318

**Part 1: Image Classification**

**Dataset Description**

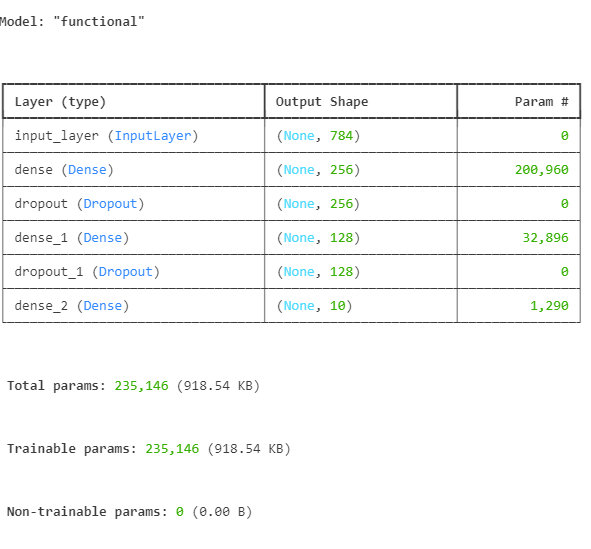
We used two datasets for image classification:

1. **Fashion-MNIST** – 28x28 grayscale images of clothing items, categorized into 10 classes (e.g., t-shirt, coat, sneaker).
2. **CIFAR-10** – 32x32 RGB images across 10 object classes (e.g., airplane, cat, car).

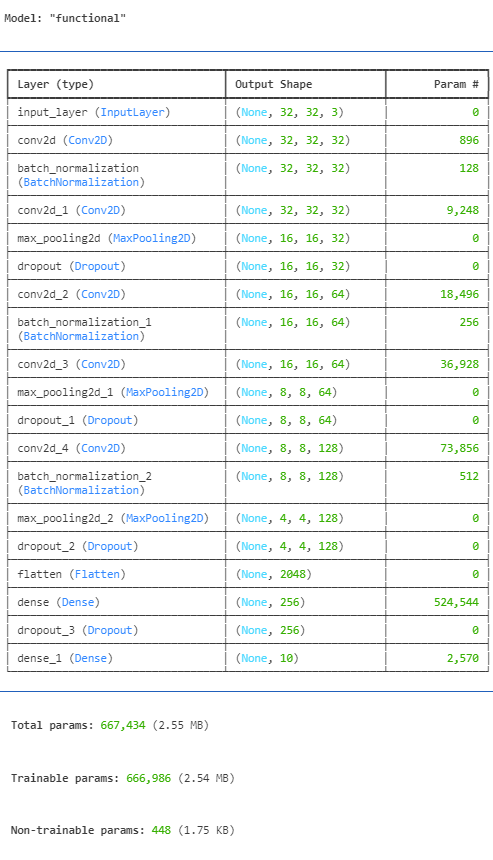
Both datasets are available through tensorflow.keras.datasets.

**Architectures**

* **Fashion-MNIST (MLP)**: A fully connected neural network using the Functional API with two hidden layers (256 and 128 units), ReLU activation, Dropout (0.3), and Glorot uniform initialization.



* **CIFAR-10 (CNN)**: A convolutional neural network with three convolution blocks followed by flattening and dense layers. Each block contains Conv2D layers, BatchNormalization, MaxPooling, and Dropout. Training used EarlyStopping.



**Training/Tuning Process**

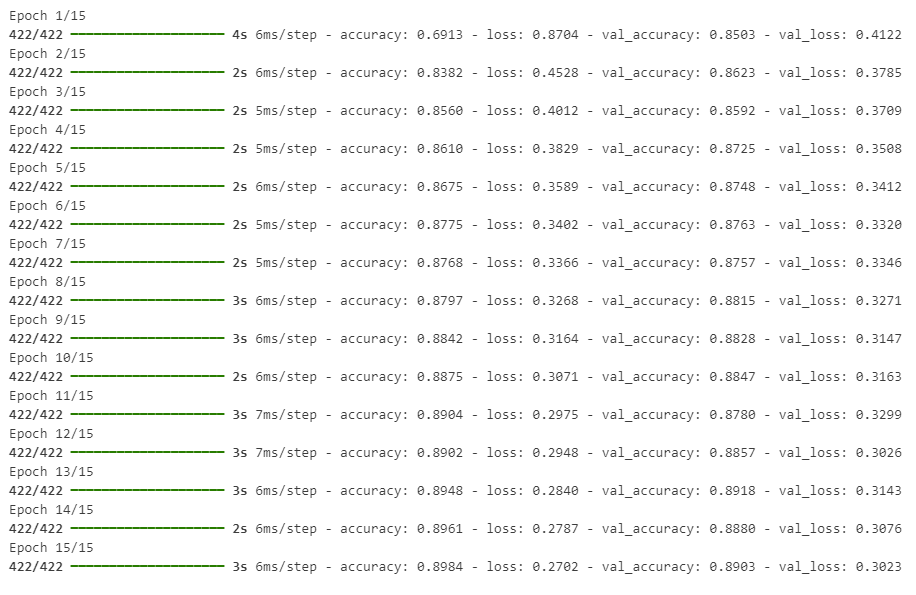
* All models used the Adam optimizer.
* Early stopping was applied with patience of 3 epochs.
* Training included 10–30 epochs with batch size 128.
* Dropout and BatchNormalization were added to prevent overfitting and stabilize training.

**Challenges & Solutions**

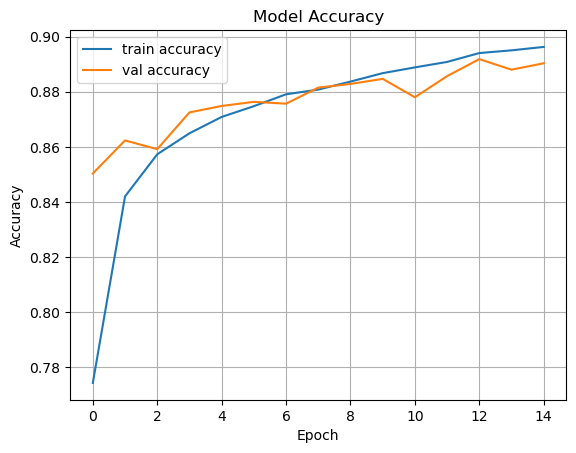
* CIFAR-10 CNN initially had slow learning. We resolved this by deepening the architecture and adding BatchNormalization and Dropout.
* Overfitting was observed in MLP models beyond epoch 10. EarlyStopping was used to handle this.

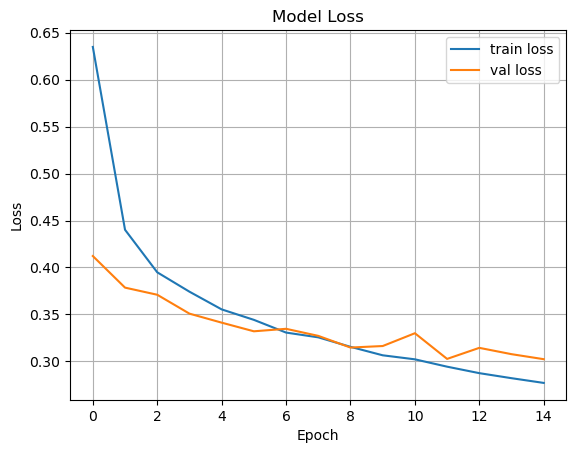
**Experimental Results**

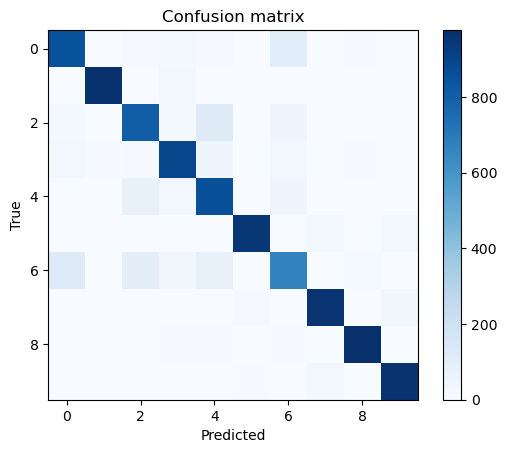
* **Fashion-MNIST (MLP)**:
  + Test accuracy: **88.7%**
  + Validation accuracy peaked at **89.0%**





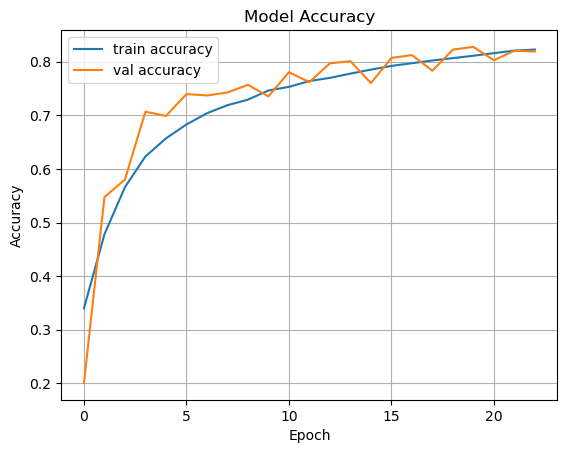


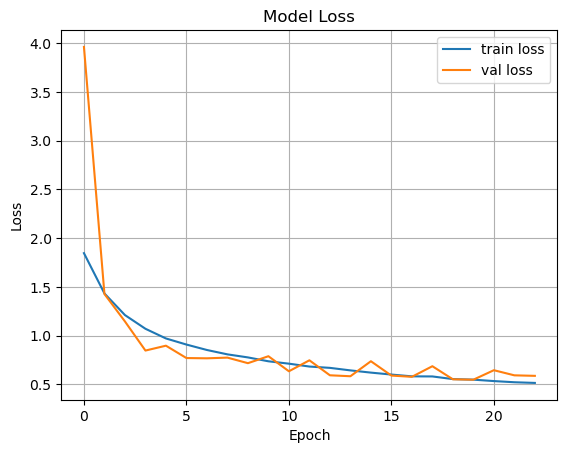


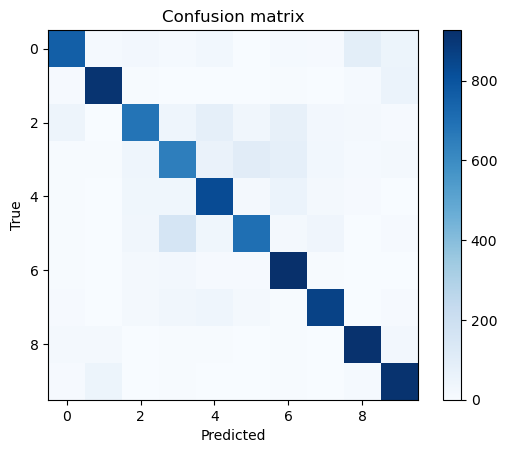


* **CIFAR-10 (CNN)**: Accuracy plateaued around 71% on test set (improved over baseline)









**Part 2: X-ray Classification – MURA Dataset**

**Strategy & Implementation Report: MURA Dataset – Multitask CNN & Transfer Learning**

**1. Project Strategy**

The objective of this project is to build an efficient deep learning pipeline for the MURA dataset (musculoskeletal radiographs), focusing on:

* **Binary classification**: Detecting normal vs. abnormal cases.
* **Multitask learning**: Simultaneously predicting the body part category.
* **Efficiency**: Optimizing model training and inference performance on Google Colab, with a modular structure for reuse and compatibility across environments.

**Key Design Decisions:**

* **Multitask architecture**: A second output branch was added to the model to predict the body part, enhancing generalization and acting as implicit attention.
* **Inclusion of an "Other" class**: A general body part class was introduced to capture rare or unknown categories, improving robustness.
* **Use of pretrained models**: EfficientNetB0 was selected for its balance between performance and computational cost.
* **Colab optimization**: Colab Pro was used to leverage GPU acceleration and increased RAM availability.
* **Modular design**: Code was organized into reusable modules (e.g., data\_loader.py, model\_pretrained.py, metrics.py) to maintain clarity and flexibility.
* **Keras Tuner integration**: Included for future hyperparameter tuning.

**2. Implementation Steps**

1. **Data loading and preprocessing**  
   A caching mechanism was implemented to preprocess the dataset locally, save it to disk, and later load it efficiently in Colab. This minimized redundant computation and optimized RAM usage.
2. **Model development**  
   Two models were implemented: a CNN from scratch and a pretrained EfficientNetB0 model with a shared backbone and dual-task outputs (binary classification and body part prediction).
3. **Model evaluation**  
   The evaluation includes F1 score, accuracy, precision, and recall for the binary classification task (normal vs. abnormal), as well as categorical accuracy for the body part prediction.

**3. Challenges and Solutions**

| **Challenge** | **Solution** |
| --- | --- |
| Slow preprocessing time on Colab | Preprocessed the dataset locally and stored it using joblib for faster loading. |
| RAM limitations in Colab | Used memory-efficient data representations and batch training. |
| TensorFlow installation issues locally | Downgraded Python to version 3.10 to ensure compatibility with required packages. |
| Class imbalance and distribution across body parts | Used multitask learning to encourage shared representations across categories. |
| Need for modularity and reproducibility | Structured code into independent modules and centralized configuration. |
| Monitoring data loading progress | Incorporated tqdm to visualize progress during preprocessing. |
| Differences in runtime environments | Standardized all configurations and file paths using a shared config.py module. |

**4. Results and Reflections**

With the implemented approach:

* The full dataset can be preprocessed and cached locally, significantly reducing Colab runtime.
* Training pipelines for both CNN and pretrained models are functional, efficient, and reusable.
* The multitask learning setup supports better feature extraction and performance consistency.
* The modular codebase enables easy experimentation, extension, and deployment.

This approach aligns with best practices in both academic research and real-world machine learning development.

**Submission Package**

* ✅ Clean .ipynb notebooks for all parts
* ✅ Report (PDF and Markdown/Notebook)
* ✅ Screenshots: model training logs, plots, confusion matrices
* ✅ GitHub link: [<https://github.com/jasproudis/deep-learning-assignment>]

*This report documents the experiments and results obtained from applying deep learning techniques to classification problems involving both common image datasets and medical radiographs.*