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

Name: Iason-Christoforos Asproudis  
Student ID: p3352318

**Part 1: Image Classification**

**Dataset Description**

We utilized two datasets for initial image classification experiments:

* *Fashion-MNIST*: 28x28 grayscale images categorized into 10 clothing item classes.
* *CIFAR-10*: 32x32 RGB images across 10 object classes such as airplanes, cats, and cars.

Both datasets are provided through tensorflow.keras.datasets.

**Model Architectures**

**Fashion-MNIST (MLP):**

* Fully connected neural network using Functional API
* Two hidden layers with 256 and 128 units
* ReLU activation and Dropout (0.3)
* Glorot uniform initialization

**CIFAR-10 (CNN):**

* Three convolutional blocks with Conv2D, BatchNormalization, MaxPooling, and Dropout
* Flatten and dense layers at the end
* Trained with EarlyStopping

**Training Strategy**

* Optimizer: Adam
* EarlyStopping with patience of 3
* Training epochs: 10–30
* Batch size: 128
* Dropout and BatchNormalization to reduce overfitting and stabilize training

**Challenges & Solutions**

* Slow convergence on CIFAR-10: Deepened architecture and added normalization layers
* Overfitting in Fashion-MNIST: Resolved with EarlyStopping and Dropout

**Results Summary**

**Fashion-MNIST (MLP):**

* Test Accuracy: *88.7%*
* Validation Accuracy: *89.0%*

**CIFAR-10 (CNN):**

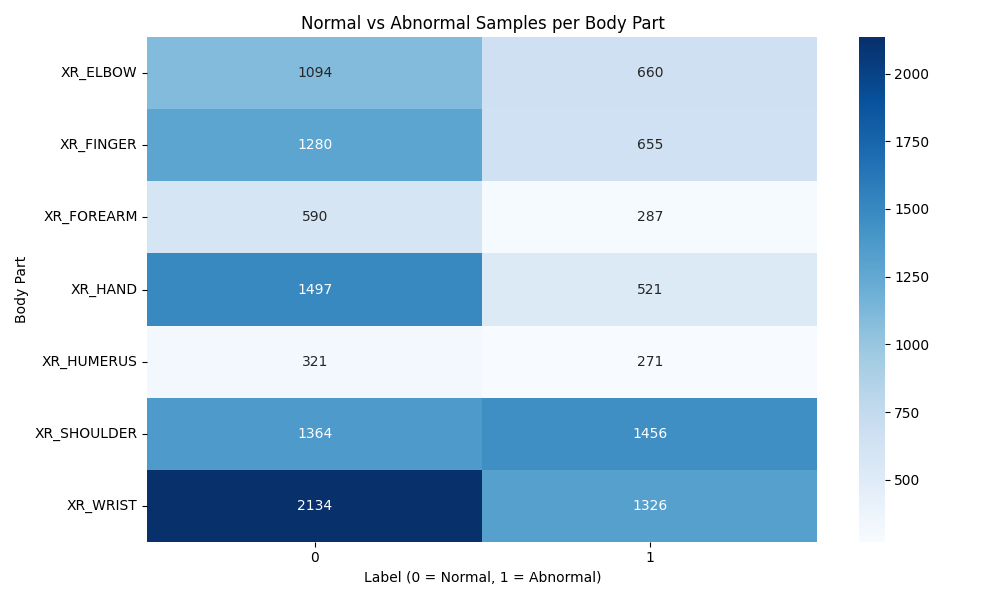
* Test Accuracy: *~71%*

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

**Dataset Overview**

The MURA dataset includes musculoskeletal radiographs categorized as either *normal* or *abnormal* across seven body parts. An analysis of the training split revealed:

* **Total labeled studies:** 13,456
* **Normal (label 0):** 8280 (61.5%)
* **Abnormal (label 1):** 5176 (38.5%)



Body parts are unevenly represented, with **wrist** and **shoulder** dominating the dataset. Notably, **shoulder** is the only body part where abnormal cases outnumber normal ones. This imbalance was visualized using a heatmap and informed our training strategy.

**Learning Approach**

The primary objective was to develop an efficient, flexible, and high-performing deep learning pipeline capable of:

* **Binary classification:** Normal vs. Abnormal
* **Multitask learning:** Predicting the body part as a secondary task
* **Modular design:** Each component implemented in reusable files (e.g., data\_loader.py, model\_pretrained.py)
* **Hardware optimization:** Designed to run smoothly on Colab Pro (A100 GPU), avoiding memory overload
* **Generalization:** Addressed class imbalance using sample weights and data augmentation

**Model Development**

**CNN from Scratch**

* Custom multitask CNN architecture with two output heads (binary + body part)
* Batch normalization and dropout for regularization
* Metrics: binary accuracy, F1-score, precision, recall, categorical accuracy

**EfficientNetB0 (Transfer Learning)**

* Backbone initialized with ImageNet weights (include\_top=False)
* Applied global average pooling and sigmoid head
* Two-phase training:
  + *Frozen base:* Train custom head only
  + *Fine-tuning:* Unfreeze top layers from index 150 and retrain with a low LR (1e-5)
* Image input: resized to *244x244 RGB*, normalized to *[0, 1]*

**Data Handling**

* **Preprocessing:**
  + Grayscale images converted to RGB
  + Cached datasets locally to speed up loading
  + Reduced memory using float16 precision
* **Augmentation:**
  + Applied flipping, brightness, and contrast variations during training
  + Significantly improved generalization on the abnormal class
* **Class Imbalance:**
  + Used compute\_sample\_weight from sklearn to assign instance-level weights

**Results**

**CNN from Scratch (Multitask)**

* **Binary Accuracy:** 64.9%
* **F1 Score (Abnormal):** 0.60
* **Body Part Accuracy:** ~100%
* **Precision (Abnormal):** 0.66
* **Recall (Abnormal):** 0.56

**EfficientNetB0 with Augmentation**

* **Binary Accuracy:** ~91%
* **Precision:** ~95%
* **Recall:** ~95%
* **F1 Score (Abnormal):** ~0.95

EfficientNetB0 showed a significant performance boost, especially in detecting abnormal cases, when combined with augmentation and fine-tuning.

**Challenges and Strategy Shifts**

**Challenges Encountered**

* A100 GPU memory exceeded during multitask training
* Shape mismatch in multitask output with pretrained models
* Runtime crashes in Colab due to RAM overflow
* Input format issues: grayscale vs RGB
* Class weight imbalance affected learning stability
* TensorFlow/Keras version incompatibilities
* Layer mismatch when mixing inputs in Functional API
* Overfitting with frozen pretrained models
* File path inconsistencies between local and cloud

**Strategies and Fixes**

* Switched from multitask to binary-only training for EfficientNet
* Used float16 and reduced batch sizes to manage memory
* Migrated from ImageDataGenerator to tf.data.Dataset with caching
* Replaced class weights with sample\_weight for better granularity
* Converted grayscale images to RGB and resized appropriately
* Used augmentation to improve generalization
* Applied EarlyStopping and ReduceLROnPlateau
* Used ModelCheckpoint with timestamps to save best models
* Visualized label distributions to guide augmentation

**Final Submission Checklist**

* ✅ Clean and modular .ipynb notebooks
* ✅ Complete report with visualizations and metrics
* ✅ Preprocessing modules and training scripts
* ✅ GitHub repository: [<https://github.com/jasproudis/deep-learning-assignment>]