Deep Learning Assignment Report – 2024-25

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 (Figure 1)and informed our training strategy.

Learning Approach

The pipeline was designed to support:

- Binary classification: Normal vs. Abnormal

- Multitask learning: Body part prediction as auxiliary task

- Modular design with reusable code components

- Resource optimization for Colab Pro (A100 GPU)

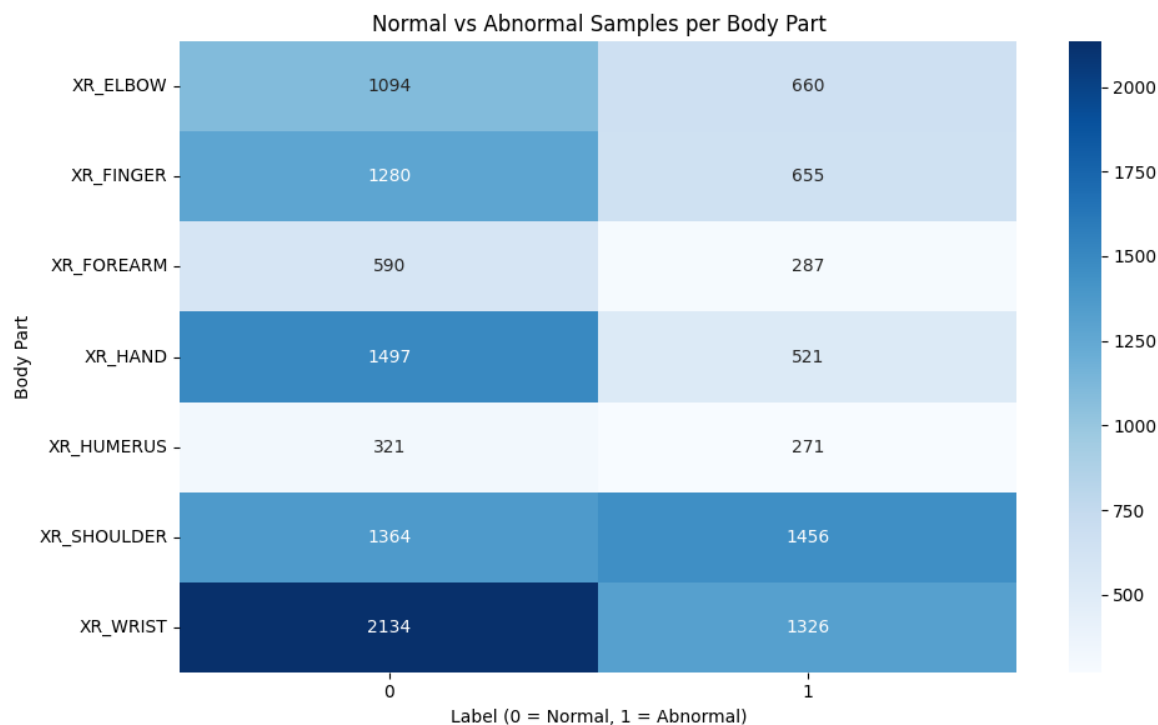- Generalization using data augmentation and sample weighting

Figure 1: heatmap of MURA Dataset

Model Development

CNN from Scratch:

- Custom multitask CNN with two output heads
- Batch normalization and dropout
- Metrics: accuracy, F1-score, precision, recall, categorical accuracy

EfficientNetB0 (Transfer Learning):

- Pretrained on ImageNet with global average pooling
- Two-phase training: frozen + fine-tuned (top layers)
- Inputs resized to 244x244 RGB and normalized to [0, 1]

Data Handling

- Grayscale to RGB conversion
- Local caching and float16 usage for memory efficiency
- Data augmentation: flip, brightness, contrast
- Sample weights using `compute_sample_weight`

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

Challenges and Strategy Shifts

Challenges Encountered:

- GPU memory exceeded with multitask models

- Output shape mismatch in pretrained multitask architectures

- Crashes in Colab from RAM overflow

- Grayscale inputs incompatible with EfficientNetB0

- Class weight instability in multitask setup

- TensorFlow/Keras version incompatibilities

- Functional API input mismatches

- Overfitting with frozen base models

- Path inconsistencies between local and Colab

Solutions Implemented:

- Switched to binary-only for pretrained models

- Used float16 and smaller batch sizes

- Replaced `ImageDataGenerator` with `tf.data.Dataset`

- Applied `sample_weight` over `class_weight`

- Data augmentation for robustness

- EarlyStopping and ReduceLROnPlateau

- Saved best checkpoints with timestamps

- Guided data decisions using heatmaps

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`