Deep Learning Final Project Documentation

Title Page

Project Title: Deep Learning Final Project - CNN and RNN Implementations

Author: Mohamed Waleed Sharshar

GitHub Repository: https://github.com/mohamedsharshar/FINAL_DEEP_LEARN

Field: Deep Learning / Artificial Intelligence

Course: Neural Network and Deep Learning

Date: December 2025

Abstract

This project presents a comprehensive implementation of two fundamental deep learning architectures: Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN/LSTM). The project addresses two distinct real-world problems:

1. **Pneumonia Detection from Chest X-Ray Images:** A CNN-based binary classification system that analyzes chest X-ray images to distinguish between normal and pneumonia-affected cases, contributing to medical AI research and automated diagnosis.
2. **Emoji Prediction from Text (Sentiment Analysis):** An LSTM-based natural language processing system that predicts appropriate emojis for English sentences, demonstrating sequence modeling and sentiment analysis capabilities.

Both implementations showcase modern deep learning techniques including data augmentation, transfer learning with pre-trained embeddings, regularization strategies, and deployment-ready API development.

1. Introduction

1.1 Background on Deep Learning

Deep learning represents a subset of machine learning that utilizes artificial neural networks with multiple layers (deep architectures) to learn hierarchical representations of data. These networks have revolutionized various fields including computer vision, natural language processing, speech recognition, and medical imaging.

The key advantage of deep learning lies in its ability to automatically learn features from raw data, eliminating the need for manual feature engineering. This capability has led to breakthrough performances in tasks such as image classification, object detection, machine translation, and sentiment analysis.

1.2 Project Motivation

The motivation behind this project stems from the desire to apply theoretical deep learning concepts to practical, real-world problems:

1. **Medical Imaging (CNN):** Pneumonia remains a leading cause of death worldwide, particularly in developing countries. Automated detection systems can assist healthcare professionals in faster and more accurate diagnosis, especially in resource-limited settings.
2. **Natural Language Processing (RNN/LSTM):** Understanding sentiment and context in text is crucial for human-computer interaction. Emoji prediction serves as an accessible demonstration of how machines can understand and respond to human emotions expressed in text.

1.3 Project Objectives

- Implement a CNN architecture for binary image classification (pneumonia detection)
- Develop an LSTM-based model for sequence classification (emoji prediction)
- Apply modern deep learning techniques including data augmentation, regularization, and transfer learning
- Create deployment-ready applications with REST APIs
- Document the complete development process for educational purposes

2. Theoretical Background

2.1 Convolutional Neural Networks (CNN)

2.1.1 Architecture Overview

CNNs are specialized neural networks designed for processing structured grid data, particularly images. The architecture consists of:

Convolutional Layers: Apply learnable filters to extract local features. Each filter detects specific patterns (edges, textures, shapes) through the convolution operation:

```
Output[i,j] = Σ Σ Input[i+m, j+n] × Kernel[m,n] + bias
```

Pooling Layers: Reduce spatial dimensions while retaining important features. Max pooling selects the maximum value in each region, providing translation invariance.

Fully Connected Layers: Combine extracted features for final classification. These layers learn non-linear combinations of high-level features.

2.1.2 Key Components Used in This Project

Component	Purpose	Implementation
Batch Normalization	Stabilize training, reduce internal covariate shift	After each conv layer
Dropout	Prevent overfitting by randomly deactivating neurons	25% in conv, 50% in FC
ReLU Activation	Introduce non-linearity, avoid vanishing gradients	f(x) = max(0, x)
Sigmoid Output	Binary classification probability	σ(x) = 1/(1+e^(-x))

2.2 Recurrent Neural Networks (RNN) and LSTM

2.2.1 RNN Fundamentals

RNNs process sequential data by maintaining a hidden state that captures information from previous time steps:

```
h_t = tanh(W_hh × h_{t-1} + W_xh × x_t + b_h)
y_t = W_hy × h_t + b_y
```

However, standard RNNs suffer from the vanishing gradient problem, making it difficult to learn long-term dependencies.

2.2.2 LSTM Architecture

Long Short-Term Memory (LSTM) networks address this limitation through a gating mechanism:

Forget Gate: Decides what information to discard from the cell state

```
f_t = σ(W_f × [h_{t-1}, x_t] + b_f)
```

Input Gate: Determines what new information to store

```
i_t = σ(W_i × [h_{t-1}, x_t] + b_i)
C̃t = tanh(W_C × [h_{t-1}, x_t] + b_C)
```

Cell State Update: Combines old and new information

```
C_t = f_t × C_{t-1} + i_t × C̃t
```

Output Gate: Controls the output based on cell state

```
o_t = σ(W_o × [h_{t-1}, x_t] + b_o)
h_t = o_t × tanh(C_t)
```

2.3 Word Embeddings (GloVe)

GloVe (Global Vectors for Word Representation) creates dense vector representations of words by analyzing word co-occurrence statistics from large text corpora. Key properties:

- **Semantic Similarity:** Words with similar meanings have similar vectors
- **Analogical Reasoning:** vector("king") - vector("man") + vector("woman") ≈ vector("queen")
- **Dimensionality:** 50-dimensional vectors capture rich semantic information

---

3. Project 1: Pneumonia Detection using CNN

3.1 Problem Statement

Develop an automated system to classify chest X-ray images as either Normal or Pneumonia-affected, assisting medical professionals in diagnosis.

3.2 Dataset Description

Source: keremberke/chest-xray-classification from HuggingFace

Split	Samples	Purpose
Training	4,077	Model learning
Validation	1,165	Hyperparameter tuning
Test	582	Final evaluation

Classes:

- Class 0: NORMAL (healthy lungs)
- Class 1: PNEUMONIA (infected lungs)

3.3 Data Preprocessing

1. **Image Resizing:** All images resized to 128×128 pixels
2. **Grayscale Conversion:** Single channel input for computational efficiency
3. **Normalization:** Pixel values scaled to [0, 1] range
4. **Data Augmentation:** Applied to training data only
    - Random rotation (±15°)
    - Horizontal flip
    - Random affine transformations

3.4 Model Architecture

```
PneumoniaCNN(
  (conv_layers): Sequential(
    # Block 1
    Conv2d(1, 32, kernel_size=3, padding=1)
    BatchNorm2d(32)
    ReLU()
    MaxPool2d(2, 2)
    Dropout(0.25)

    # Block 2
    Conv2d(32, 64, kernel_size=3, padding=1)
    BatchNorm2d(64)
    ReLU()
    MaxPool2d(2, 2)
    Dropout(0.25)

    # Block 3
    Conv2d(64, 128, kernel_size=3, padding=1)
    BatchNorm2d(128)
    ReLU()
    MaxPool2d(2, 2)
    Dropout(0.25)

    # Block 4
    Conv2d(128, 256, kernel_size=3, padding=1)
    BatchNorm2d(256)
    ReLU()
    MaxPool2d(2, 2)
    Dropout(0.25)
  )

  (fc_layers): Sequential(
    Linear(16384, 256)
    ReLU()
    Dropout(0.5)
    Linear(256, 128)
    ReLU()
    Dropout(0.5)
    Linear(128, 1)
    Sigmoid()
  )
)
```

Total Parameters: ~4.3 million

3.5 Training Configuration

Parameter	Value	Justification
Optimizer	Adam	Adaptive learning rates, momentum
Learning Rate	0.001	Standard starting point
Loss Function	Binary Cross Entropy	Binary classification task
Batch Size	32	Balance between speed and stability
Epochs	25	With early stopping (patience=5)
LR Scheduler	ReduceLROnPlateau	Reduce LR when validation loss plateaus

3.6 Handling Class Imbalance

The dataset exhibits class imbalance. To address this:

1. **Weighted Random Sampling:** Oversample minority class during training
2. **Class Weights:** Apply higher penalty for misclassifying minority class

```
class_weights = 1.0 / class_counts
sample_weights = class_weights[labels]
sampler = WeightedRandomSampler(sample_weights, len(sample_weights))
```

3.7 Evaluation Metrics

- **Accuracy:** Overall correct predictions
- **Precision:** True positives / (True positives + False positives)
- **Recall (Sensitivity):** True positives / (True positives + False negatives)
- **F1-Score:** Harmonic mean of precision and recall
- **AUC-ROC:** Area under the Receiver Operating Characteristic curve

3.8 Results and Analysis

The model demonstrates strong performance in detecting pneumonia from chest X-rays:

- High sensitivity ensures most pneumonia cases are detected
- Balanced precision-recall trade-off minimizes false alarms
- Confusion matrix analysis reveals model behavior across classes

---

4. Project 2: Emoji Prediction using LSTM

4.1 Problem Statement

Build a sentiment analysis system that predicts the most appropriate emoji for a given English sentence, demonstrating sequence modeling capabilities.

4.2 Dataset Description

Training Data: 132 labeled sentences **Test Data:** 56 sentences

Emoji Classes:

Class	Emoji	Meaning	Example Sentence
0	❤	Love	"I love you"
1	⚾	Sports	"Let's play baseball"
2	😄	Happy	"I am so happy"
3	😞	Sad	"I feel terrible"
4	🍴	Food	"Let's eat dinner"

4.3 Word Embeddings

GloVe 6B 50d: Pre-trained on 6 billion tokens from Wikipedia and Gigaword

- Vocabulary size: 400,001 words
- Embedding dimension: 50
- File size: ~170 MB

Embedding Matrix Construction:

```
def read_glove_vecs(glove_file):
    word_to_vec_map = {}
    with open(glove_file, encoding="utf-8") as f:
        for line in f:
            values = line.split()
            word = values[0]
            vec = np.array(values[1:], dtype=np.float64)
            word_to_vec_map[word] = vec
    return word_to_vec_map
```

4.4 Model Architecture

```
Model: "Emojify_LSTM"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 10, 50)            20,000,050
lstm_1 (LSTM)                (None, 10, 128)           91,648
dropout_1 (Dropout)          (None, 10, 128)           0
lstm_2 (LSTM)                (None, 128)               131,584
dropout_2 (Dropout)          (None, 128)               0
dense (Dense)                (None, 5)                 645
=================================================================
Total params: 20,223,927
Trainable params: 223,877 (embeddings frozen)
```

4.5 Training Process

1. **Sentence Preprocessing:**
    - Convert to lowercase
    - Tokenize into words
    - Pad/truncate to max length (10 words)
    - Convert words to indices
2. **Training Configuration:**

Parameter	Value
Max Sequence Length	10
LSTM Units	128 (2 layers)
Dropout Rate	0.5
Optimizer	Adam
Loss Function	Categorical Cross Entropy
Epochs	100
Batch Size	32

4.6 Results

```
Training Accuracy: ~98.5%
Test Accuracy: ~87.5%
```

The model successfully learns to associate sentence semantics with appropriate emojis.

4.7 Deployment

FastAPI REST API

```
@app.post("/predict")
async def predict_emoji(request: TextRequest):
    text = request.text
    emoji, confidence, all_preds = predict_single(text)
    return {
        "text": text,
        "emoji": emoji,
        "confidence": confidence,
        "all_predictions": all_preds
    }
```

Docker Deployment

```
FROM python:3.10-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY . .
EXPOSE 7860
CMD ["python", "app.py"]
```

Hugging Face Spaces

The model is deployed on Hugging Face Spaces for public access and demonstration.

---

5. Implementation Details

5.1 Development Environment

- **Programming Language:** Python 3.8+
- **Deep Learning Frameworks:** PyTorch (CNN), TensorFlow/Keras (LSTM)
- **Data Processing:** NumPy, Pandas
- **Visualization:** Matplotlib, Seaborn
- **API Framework:** FastAPI
- **Containerization:** Docker

5.2 Project Structure

```
FINAL_DEEP_LEARN/
├── Final_Project_CNN/
│   ├── pneumonia_detection_notebook_documented_CNN.ipynb
│   ├── README.md
│   └── README_Arabic.md
│
├── Final_Project_RNN/
│   ├── main.py                # Training script
│   ├── app.py                 # FastAPI server
│   ├── emo_utils.py           # Helper functions
│   ├── train_emoji.csv        # Training data
│   ├── test_emoji.csv         # Test data
│   ├── glove.6B.50d.txt       # Word embeddings
│   ├── model.weights.h5       # Trained weights
│   ├── requirements.txt       # Dependencies
│   ├── Dockerfile             # Container config
│   ├── README.md
│   └── README_Arabic.md
│
└── Deep_Learning_Project_Documentation.md
```

5.3 Key Code Snippets

CNN Training Loop

```
def train_epoch(model, dataloader, criterion, optimizer, device):
    model.train()
    running_loss = 0.0
    correct = 0
    total = 0

    for images, labels in dataloader:
        images, labels = images.to(device), labels.to(device)

        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs.squeeze(), labels.float())
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
        predicted = (outputs.squeeze() > 0.5).float()
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

    return running_loss / len(dataloader), correct / total
```

LSTM Model Definition

```
def Emojify_V2(input_shape, word_to_vec_map, word_to_index):
    sentence_indices = Input(shape=input_shape, dtype='int32')

    embedding_layer = pretrained_embedding_layer(word_to_vec_map, word_to_index)
    embeddings = embedding_layer(sentence_indices)

    X = LSTM(128, return_sequences=True)(embeddings)
    X = Dropout(0.5)(X)
    X = LSTM(128)(X)
    X = Dropout(0.5)(X)
    X = Dense(5, activation='softmax')(X)

    model = Model(inputs=sentence_indices, outputs=X)
    return model
```

---

6. Challenges and Solutions

6.1 CNN Project Challenges

Challenge	Solution
Class imbalance	Weighted random sampling, class weights
Overfitting	Dropout, data augmentation, early stopping
Limited GPU memory	Batch size optimization, gradient accumulation
Slow convergence	Learning rate scheduling, batch normalization

6.2 RNN Project Challenges

Challenge	Solution
Small dataset	Pre-trained GloVe embeddings (transfer learning)

Challenge	Solution
Variable sentence length	Padding to fixed length (10 words)
Vanishing gradients	LSTM architecture with gating mechanism
Out-of-vocabulary words	Zero vector for unknown words

---

7. Conclusions

7.1 Summary

This project successfully demonstrates the implementation of two fundamental deep learning architectures:

1. **CNN for Medical Imaging:** The pneumonia detection model achieves strong performance in classifying chest X-rays, showcasing the potential of AI in healthcare applications.
2. **LSTM for NLP:** The emoji prediction system effectively captures sentence semantics and maps them to appropriate emotional representations.

7.2 Key Learnings

- Importance of data preprocessing and augmentation
- Effectiveness of transfer learning (GloVe embeddings)
- Role of regularization in preventing overfitting
- Value of proper evaluation metrics for imbalanced datasets
- Practical deployment considerations (API, Docker, cloud)

7.3 Future Work

- **CNN:** Explore transfer learning with pre-trained models (ResNet, VGG)
- **LSTM:** Experiment with attention mechanisms and transformers
- **Both:** Implement model interpretability techniques (Grad-CAM, attention visualization)
- **Deployment:** Add model versioning and A/B testing capabilities

---

8. References

1. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.
2. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.
3. Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. EMNLP.
4. Rajpurkar, P., et al. (2017). CheXNet: Radiologist-level pneumonia detection on chest X-rays with deep learning.
5. PyTorch Documentation: https://pytorch.org/docs/
6. TensorFlow/Keras Documentation: https://www.tensorflow.org/api_docs
7. FastAPI Documentation: https://fastapi.tiangolo.com/

Appendix A: Installation Guide

A.1 CNN Project Setup

```
# Clone repository
git clone https://github.com/mohamedsharshar/FINAL_DEEP_LEARN.git
cd FINAL_DEEP_LEARN/Final_Project_CNN

# Install dependencies
pip install datasets torch torchvision tqdm scikit-learn seaborn matplotlib pillow

# Run notebook
jupyter notebook pneumonia_detection_notebook_documented_CNN.ipynb
```

A.2 RNN Project Setup

```
cd Final_Project_RNN

# Install dependencies
pip install -r requirements.txt

# Download GloVe embeddings
# From: https://nlp.stanford.edu/data/glove.6B.zip
# Extract glove.6B.50d.txt to project folder

# Train model
python main.py

# Run API
python app.py
```

Appendix B: API Documentation

B.1 Emoji Predictor API

Base URL: `http://localhost:7860`

Health Check

```
GET /health
Response: {"status": "healthy"}
```

Single Prediction

```
POST /predict
Body: {"text": "I love you"}
Response: {
    "text": "I love you",
    "emoji": "❤",
    "emoji_meaning": "love",
    "confidence": 0.95,
    "all_predictions": {
        "❤": 0.95,
        "😄": 0.03,
        "😞": 0.01,
        "": 0.005,
        "": 0.005
    }
}
```

Batch Prediction

```
POST /predict/batch
Body: ["I love you", "Let's eat pizza", "I am sad"]
Response: [
    {"text": "I love you", "emoji": "❤", ...},
    {"text": "Let's eat pizza", "emoji": "", ...},
    {"text": "I am sad", "emoji": "😞", ...}
]
```

Document Version: 1.0 **Last Updated:** December 2025 **Author:** Mohamed Waleed Sharshar