CNN Assignment: Student Instructions

Convolutional Neural Networks for Image Classification

AIMLCZG511 Deep Neural Networks Course
BITS Pilani Work Integrated Learning Programme

Semester 1, 2025-26

CRITICAL - READ CAREFULLY

Submission Deadline: 08-Feb-26
Total Marks: 20 (Scaled to 10)
Submission Attempts: ONE only
Submission Mode: LMS only (NO email submissions)
File Format: .ipynb ONLY (NO zip files, NO data files)

1 Assignment Overview

1.1 Learning Objectives

By completing this assignment, you will:

- Design CNN architectures using industry-standard frameworks (Keras/PyTorch)
- Implement Global Average Pooling for reducing overfitting
- Apply transfer learning using pre-trained models (ResNet/VGG)
- Compare custom CNN vs transfer learning approaches
- Evaluate models using classification metrics

1.2 What You Will Build

1. **Custom CNN:** Build a CNN from scratch using Keras or PyTorch
2. **Transfer Learning Model:** Fine-tune a pre-trained ResNet or VGG model
3. **Comparison:** Analyze performance, training time, and complexity

1.3 Time Estimate

Expected Time: 6-8 hours total

- Dataset preparation: 1-2 hours
- Custom CNN implementation: 2-3 hours
- Transfer learning: 1-2 hours
- Analysis and documentation: 1 hour

2 Submission Requirements

CRITICAL - READ CAREFULLY

AUTOMATIC ZERO MARKS if ANY of these are violated:

1. Wrong filename format
2. BITS ID mismatch (filename vs notebook)
3. Student name mismatch (LMS Name vs notebook)
4. Notebook not executed (no outputs visible)
5. Execution errors present
6. Using Flatten+Dense instead of Global Average Pooling
7. Missing JSON output
8. File submitted as zip or with separate data files

2.1 Filename Format

Required format: `<BITS_ID>_rnn_assignment.ipynb`

Examples:

- ✓ `2025AA05036_rnn_assignment.ipynb`
- ✗ `RNN_Assignment.ipynb`
- ✗ `2025AA05036_Assignment3.ipynb`
- ✗ `TimeSeriesProject.ipynb`

2.2 Student Information

Fill in the first cell of your notebook:

```
BITS ID: 2025AA01234
Name: JOHN DOE
Email: john.doe@wilp.bits-pilani.ac.in
Date: 15-01-2026
```

CRITICAL: BITS ID in filename MUST match BITS ID in notebook.

2.3 Execution Requirements

Before submission:

1. Click **Kernel → Restart & Run All**
2. Wait for ALL cells to execute
3. Verify ALL outputs are visible
4. Check for any error messages
5. Ensure JSON output is printed at the end

2.4 File Submission

- Submit ONLY the `.ipynb` file
- NO zip files
- NO separate data files or image folders
- NO Python scripts (.py files)
- All code and outputs must be in the notebook

3 Technical Requirements

3.1 Framework Choice

Choose ONE framework (do NOT mix):

- **Option 1:** Keras/TensorFlow
- **Option 2:** PyTorch

  Both are equally acceptable. Use the one you're comfortable with.

3.2 Dataset Requirements

3.2.1 Allowed Datasets

Choose ONE from:

1. **Cats vs Dogs** (2 classes, ∼25,000 images)
2. **Food-101 subset** (10-20 classes, subset of original)
3. **Plant Disease** (3-5 classes, agricultural images)
4. **Medical Images** (X-rays, CT scans - 2-3 classes)
5. **Custom dataset** (requires instructor approval)

3.2.2 Dataset Specifications

- **Minimum:** 500 images **PER CLASS** (not total)
- **Classes:** 2-20 classes
- **Train/Test Split:** 90/10 OR 85/15 (choose one)
- **Image Size:** Typically 224×224 for transfer learning

3.3 Global Average Pooling (GAP) - MANDATORY

CRITICAL - READ CAREFULLY

BOTH models MUST use Global Average Pooling.
NOT ALLOWED:

```
# WRONG - will result in 0 marks
model.add(Flatten())
model.add(Dense(256))
model.add(Dense(n_classes, activation='softmax'))
```

REQUIRED:

```
# CORRECT - Keras
model.add(GlobalAveragePooling2D())
model.add(Dense(n_classes, activation='softmax'))

# CORRECT - PyTorch
self.gap = nn.AdaptiveAvgPool2d(1)
self.fc = nn.Linear(channels, n_classes)
```

Why GAP?

- Reduces overfitting compared to Flatten + Dense
- Fewer parameters, more efficient
- Standard practice in modern CNNs
- Forces better spatial feature learning

4 Assignment Components

4.1 Part 1: Custom CNN (5 marks)

4.1.1 Requirements

Build a CNN using Keras or PyTorch with:

- At least 2 Conv2D layers
- Pooling layers (MaxPooling or AveragePooling)
- **Global Average Pooling (MANDATORY)**
- Output layer with Softmax activation

4.1.2 Grading Breakdown

- Architecture design with GAP: 2 marks
- Model properly compiled/configured: 1 mark
- Training completed with loss tracking: 1 mark
- All metrics calculated correctly: 1 mark

4.1.3 What to Track

- Initial loss (first epoch)
- Final loss (last epoch)
- Training time
- All 4 metrics: accuracy, precision, recall, F1-score

4.2 Part 2: Transfer Learning (5 marks)

4.2.1 Requirements

Use a pre-trained model:

- **Choose ONE:** ResNet18, ResNet50, VGG16, or VGG19
- Freeze base layers (feature extractor)
- Add **Global Average Pooling (MANDATORY)**
- Add custom classification head
- Fine-tune on your dataset

4.2.2 Grading Breakdown

- Valid base model with frozen layers: 2 marks
- GAP + custom head implementation: 1 mark
- Training completed with loss tracking: 1 mark
- All metrics calculated correctly: 1 mark

4.2.3 Architecture Documentation

You must report:

- Base model name
- Number of frozen layers
- Number of trainable layers
- Total parameters
- Trainable parameters only

4.3 Part 3: Training Process (4 marks)

4.3.1 Convergence Requirements

Loss reduction is calculated as:

$$\text{Reduction } \% = \frac{\text{Initial Loss} - \text{Final Loss}}{\text{Initial Loss}} \times 100$$

Grading:

- $\geq$50% reduction: 2 marks

- ≥20% reduction: 1 mark
- <20% reduction: 0 marks

This applies to BOTH models (2 marks each = 4 marks total).

4.4 Part 4: Metrics Calculation (2 marks)

4.4.1 Required Metrics (ALL 4 for BOTH models)

1. **Accuracy**
2. **Precision** (macro-averaged)
3. **Recall** (macro-averaged)
4. **F1-Score** (macro-averaged)

All metrics MUST be in range [0, 1].

4.4.2 Primary Metric Selection

Choose ONE as your primary metric and justify:

- **Accuracy:** For balanced datasets
- **Precision:** When false positives are costly
- **Recall:** When false negatives are costly (e.g., medical diagnosis)

Write 1-2 sentences explaining your choice based on your dataset.

4.5 Part 5: Analysis (2 marks)

4.5.1 Requirements

Write an analysis (maximum 200 words guideline) addressing:

1. Which model performed better and by how much?
2. Impact of pre-training vs training from scratch
3. Effect of Global Average Pooling
4. Computational cost comparison (time, parameters)
5. Transfer learning insights
6. Convergence behavior differences

4.5.2 Grading Criteria

- Covers 5+ key topics with depth: 2 marks
- Covers 3-4 key topics: 1 mark
- Covers <3 topics or superficial: 0 marks

Note: Word count >200 will generate a warning but NO marks deduction. Focus on *quality*, not just word count.

4.6 Part 6: Code Structure (2 marks)

4.6.1 Requirements

- Both models properly implemented: 1 mark
- JSON output with correct structure: 1 mark

5 JSON Output Format

Important Information

The autograder relies on exact JSON field names. Use the provided template's `get_assignment_results()` function WITHOUT modification.

6 Common Mistakes to Avoid

Important Information

Top reasons for losing marks:

1. Using Flatten + Dense instead of Global Average Pooling
2. Not freezing base layers in transfer learning
3. Missing metrics (only 3 out of 4)
4. Metrics outside valid range [0, 1]
5. Not tracking initial/final loss
6. JSON output missing or incorrect field names
7. Submitting with all 0.0 metric values
8. Not executing notebook before submission

6.1 Implementation Mistakes

6.1.1 Wrong: Flatten + Dense

```
model.add(Conv2D(...))
model.add(Flatten())  # WRONG
model.add(Dense(128))  # WRONG
model.add(Dense(n_classes))
```

6.1.2 Correct: Global Average Pooling

```
model.add(Conv2D(...))
model.add(GlobalAveragePooling2D())  # CORRECT
model.add(Dense(n_classes))
```

6.2 Transfer Learning Mistakes

6.2.1 Wrong: Not Freezing Layers

```
base_model = ResNet50(weights='imagenet', include_top=False)
# WRONG - forgot to freeze
```

6.2.2 Correct: Frozen Base Layers

```
base_model = ResNet50(weights='imagenet', include_top=False)
base_model.trainable = False  # CORRECT - freeze base
```

7 Helpful Tips

Helpful Tips

Before submission checklist:

1. ❑ Filename is `<BITS_ID>_cnn_assignment.ipynb`
2. ❑ BITS ID in filename matches notebook
3. ❑ Student name matches the name in LMS
4. ❑ Ran Kernel → Restart & Run All
5. ❑ All outputs visible (no blank cells)
6. ❑ No execution errors
7. ❑ Both models use Global Average Pooling
8. ❑ All 4 metrics calculated for both models
9. ❑ JSON output printed at the end
10. ❑ Analysis written (quality content)
11. ❑ Training curves plotted
12. ❑ Confusion matrices created
13. ❑ Environment screenshot included

7.1 Debugging Tips

7.1.1 If Training is Too Slow

- Reduce image size (e.g., 128×128 instead of 224×224)
- Use fewer training samples for testing
- Reduce number of epochs during development
- Use GPU if available (Google Colab provides free GPU)

7.1.2 If Accuracy is Too Low

- Check data preprocessing (normalization)
- Verify labels are correct
- Try data augmentation

- Increase number of epochs
- Adjust learning rate

7.1.3 If Memory Issues

- Reduce batch size
- Reduce image resolution
- Use fewer training samples
- Close other applications
- Use Google Colab with GPU enabled

8 Evaluation Rubric Summary

Component	Marks	Key Requirements
Custom CNN	5	GAP, compile, training, metrics
Transfer Learning	5	Base model, frozen, GAP, metrics
Training Process	4	Convergence $\geq$20% for both
Metrics	2	All 4 metrics, valid ranges
Analysis	2	Quality-based, 5+ topics
Code Structure	2	Implementations + JSON
TOTAL	**20**	**Scaled to 10**

Table 1: Marks Distribution

9 FAQs

9.1 Can I use a different framework?

No. Only Keras/TensorFlow or PyTorch are allowed.

9.2 Can I use pre-trained transformers?

No. This is a CNN assignment. Only ResNet or VGG for transfer learning.

9.3 What if my dataset is smaller?

You must have at least 500 images per class. Choose a different dataset if yours is smaller.

9.4 Can I implement CNN from scratch (pure NumPy)?

No. Use Keras or PyTorch layers. The focus is on architecture design, not low-level implementation.

9.5 Do I need to submit my dataset?

No. Submit ONLY the notebook file. Load data from public sources or URLs.

9.6 Can I resubmit if I make a mistake?

No. You have ONE submission attempt only. Test thoroughly before submitting.

9.7 Will there be partial credit?

Yes. Each section is graded independently. Even if one part fails, you get credit for completed parts.

9.8 What if my notebook doesn't open?

You will receive 0 marks. Test that your file opens correctly before submission.

10 Support and Resources

10.1 Getting Help

- **Course Forum:** Post questions on the LMS discussion board
- **Documentation:** Refer to Keras/PyTorch official docs

10.2 Useful Resources

- Keras Documentation: `https://keras.io`
- PyTorch Documentation: `https://pytorch.org`
- Transfer Learning Guide: (See course materials)

Good Luck!

Remember: Quality over quantity. Focus on correct implementation and thoughtful analysis.