Deep Learning Final Year Project Ideas PDF

A strong deep learning final year project uses a public dataset, starts from a pretrained model, compares it with a simple baseline, and explains its errors. Pneumonia detection from X-rays, plant disease classification, speech command recognition and emotion detection are proven choices that run on free Google Colab GPUs.

Key takeaways

- Use transfer learning from a pretrained network; training from scratch rarely fits a semester.
- Free Colab or Kaggle GPU time is enough for every project here.
- Show error analysis: which images, sounds or sentences the model gets wrong, and why.
- Keep a classic ML baseline so the gain from deep learning is visible.
- The PDF gives plans and steps, not finished source code.

Who this PDF is for

This PDF is for final-year B.Tech, MCA and M.Sc students who want a deep learning project with real data and a working demo. The ideas cover four data types (images, audio, text and sensor signals), so a team can pick the area its guide knows best. Each plan lists the dataset, a PyTorch or TensorFlow/Keras stack, the modules a 3 or 4 person team can split, and what examiners usually ask.

How to choose your project

- **Check GPU needs.** Pick datasets you can train on Colab in under an hour per run.
- **Start pretrained.** ResNet, EfficientNet, MobileNet or a small transformer saves weeks.
- **One data type.** Do images or audio or text well, not all three.
- **Plan the demo** (web app, mobile app or notebook) in week one.

Standard level projects

1. Pneumonia Detection from Chest X-rays

- **Problem:** Classify chest X-rays as normal or pneumonia.
- **Dataset:** Kaggle "Chest X-Ray Images (Pneumonia)".
- **Tech stack:** Python, PyTorch or Keras, torchvision, Grad-CAM, Streamlit.
- **Main modules:** Data loader with augmentation, pretrained CNN, class weighting, Grad-CAM viewer, upload app.

- **Steps:** Rebalance classes, fine-tune ResNet or DenseNet, track recall, add Grad-CAM heatmaps.
- **Show in the viva:** Grad-CAM images showing where the model looked, and a not-for-diagnosis note.
- **Extensions:** Test on a second public X-ray dataset to check generalisation.

2. Plant Leaf Disease Classifier

- **Problem:** Identify crop diseases from leaf photos.
- **Dataset:** PlantVillage dataset (available on Kaggle).
- **Tech stack:** Python, TensorFlow/Keras, MobileNetV2, TensorFlow Lite.
- **Main modules:** Augmentation, transfer learning, per-class metrics, mobile model export.
- **Steps:** Fine-tune MobileNetV2, check per-class accuracy, convert to TFLite, test on phone photos.
- **Show in the viva:** Why lab images score higher than field photos.
- **Extensions:** Add treatment tips per disease from agriculture extension sources.

3. Facial Emotion Recognition

- **Problem:** Detect emotions such as happy, sad or angry from face images.
- **Dataset:** Kaggle "FER-2013".
- **Tech stack:** Python, PyTorch, OpenCV, Flask.
- **Main modules:** Face detection, CNN classifier, confusion matrix, webcam demo.
- **Steps:** Train a small CNN baseline, fine-tune a pretrained model, compare, add live webcam inference.
- **Show in the viva:** Which emotions get confused and why.
- **Extensions:** Add a bias check across lighting conditions.

4. Speech Command Recognition

- **Problem:** Recognise short spoken words such as yes, no, up and down.
- **Dataset:** Google "Speech Commands" dataset.
- **Tech stack:** Python, PyTorch, torchaudio, librosa.
- **Main modules:** Audio loading, mel spectrograms, CNN model, noise augmentation, mic demo.
- **Steps:** Convert clips to spectrograms, train a CNN, add background noise, test on your own voice.
- **Show in the viva:** What a mel spectrogram shows.
- **Extensions:** Run the model on a Raspberry Pi.

Advanced level projects

5. Image Caption Generator

- **Problem:** Generate a sentence describing an image.
- **Dataset:** Flickr8k dataset (available on Kaggle).
- **Tech stack:** Python, PyTorch, pretrained CNN encoder, LSTM or transformer decoder.
- **Main modules:** Feature extractor, tokenizer, decoder, beam search, BLEU scoring.
- **Steps:** Extract image features, train the decoder, generate with beam search, score with BLEU.

- **Show in the viva:** Good and bad captions side by side.
- **Extensions:** Swap in a vision-language model and compare.

6. Brain Tumour MRI Classification

- **Problem:** Classify brain MRI scans as tumour or no tumour.
- **Dataset:** Kaggle "Brain MRI Images for Brain Tumor Detection".
- **Tech stack:** Python, Keras, EfficientNet, Grad-CAM.
- **Main modules:** Preprocessing (cropping, resizing), transfer learning, k-fold validation, explanation view.
- **Steps:** Crop to the brain region, augment heavily (the set is small), use k-fold, report mean and spread.
- **Show in the viva:** Why k-fold matters with a small dataset.
- **Extensions:** Add segmentation with a U-Net on a labelled set.

7. Movie Review Sentiment with LSTM and Transformer

- **Problem:** Classify reviews as positive or negative and compare architectures.
- **Dataset:** Stanford "Large Movie Review Dataset" (IMDB).
- **Tech stack:** Python, PyTorch, Hugging Face Transformers.
- **Main modules:** Tokenisation, LSTM model, fine-tuned DistilBERT, comparison report.
- **Steps:** Train an LSTM, fine-tune DistilBERT, compare accuracy, training time and model size.
- **Show in the viva:** The cost versus accuracy trade-off.
- **Extensions:** Add attention visualisation.

8. Household Energy Forecasting with LSTM

- **Problem:** Forecast next-day household power use.
- **Dataset:** UCI "Individual Household Electric Power Consumption".
- **Tech stack:** Python, Keras, pandas.
- **Main modules:** Resampling, windowing, LSTM and GRU models, naive and ARIMA baselines.
- **Steps:** Resample to hourly, build sliding windows, train LSTM and GRU, compare with baselines on MAE.
- **Show in the viva:** How window length changes error.
- **Extensions:** Try a temporal convolutional network.

Research-oriented projects

9. Environmental Sound Classification

- **Problem:** Classify sounds such as sirens, rain or dog barks.
- **Dataset:** ESC-50 dataset.
- **Tech stack:** Python, PyTorch, torchaudio.
- **Main modules:** Spectrograms, CNN, pretrained audio model, 5-fold evaluation.
- **Steps:** Use the official 5 folds, train a CNN, compare with a pretrained audio network.
- **Show in the viva:** Why the official folds must be kept.
- **Extensions:** Deploy as a noise monitor for a campus.

10. Knowledge Distillation for Mobile Models

- **Problem:** Make a small model learn from a large one with little accuracy loss.
- **Dataset:** CIFAR-10.
- **Tech stack:** Python, PyTorch.
- **Main modules:** Teacher training, student training, distillation loss, size and speed benchmarks.
- **Steps:** Train a ResNet teacher, train a small student with and without distillation, compare.
- **Show in the viva:** The accuracy, size and speed table.
- **Extensions:** Add quantisation and compare again.

11. Skin Lesion Classification

- **Problem:** Classify dermoscopy images into lesion types.
- **Dataset:** ISIC Archive images (International Skin Imaging Collaboration).
- **Tech stack:** Python, PyTorch, EfficientNet.
- **Main modules:** Class imbalance handling, transfer learning, balanced accuracy, explanation maps.
- **Steps:** Use weighted sampling, fine-tune, report balanced accuracy and per-class recall.
- **Show in the viva:** Why balanced accuracy is used here.
- **Extensions:** Add uncertainty estimates with Monte Carlo dropout.

12. Adversarial Robustness Study

- **Problem:** Test how small image changes fool a classifier, and defend against them.
- **Dataset:** Fashion-MNIST and CIFAR-10.
- **Tech stack:** Python, PyTorch.
- **Main modules:** Baseline model, FGSM and PGD attacks, adversarial training, results.
- **Steps:** Train models, attack with FGSM and PGD at several strengths, retrain with adversarial examples, compare.
- **Show in the viva:** Clean versus attacked accuracy charts.
- **Extensions:** Write it up for a student conference.

Report structure

A final-year report is marked on clarity as much as on code. A structure most universities accept:

1. **Abstract** (150 to 250 words): problem, method, main result.
2. **Introduction:** background, problem statement, objectives and scope.
3. **Literature review:** 8 to 15 papers or existing systems, and the gap your project fills.
4. **Requirements and design:** functional and non-functional requirements, architecture diagram, the network architecture diagram, input size and training settings.
5. **Implementation:** modules, tools, key algorithms and screenshots.
6. **Testing and results:** training curves, test metrics against a baseline, a confusion matrix and examples of errors.
7. **Conclusion and future scope.**
8. **References** in the format your department asks for (IEEE style is common).

Follow your own university's template if it gives one; it overrides this list.

Common mistakes to avoid

- Training from scratch on a small dataset and getting overfitting.
- Letting near-duplicate images fall into both train and test sets.
- Reporting only accuracy on imbalanced medical data.
- Not saving checkpoints, then losing a Colab session before the viva.

Frequently asked questions

Which deep learning project is best for a final year?

One with a clean public dataset, a pretrained model and a visible demo. Pneumonia detection, plant disease classification and speech command recognition are reliable and well documented.

Do I need my own GPU?

No. Free Google Colab or Kaggle notebooks give limited GPU time, which is enough for transfer learning on the datasets in this PDF.

PyTorch or TensorFlow for a final year project?

Either works. Use the one your guide or course teaches. Keras is quicker to start; PyTorch is common in research papers.

How do I explain a deep learning model in the viva?

Show the architecture, training curves, a confusion matrix and Grad-CAM or error examples. Be ready to explain transfer learning and overfitting in simple words.

Does this PDF include full source code?

No. It gives the problem, data, stack, modules and steps for each project. You write the code yourself, which is what examiners test in the viva.

Is this PDF free?

Yes. Fill in the short form on this page and the PDF downloads right away.

Sources and further reading

- PyTorch tutorials
- TensorFlow tutorials
- Kaggle Datasets
- ISIC Archive
- Hugging Face Transformers documentation

About this PDF

Prepared by the Uncodemy training team as practice project ideas. Datasets and tools named are public; check each dataset's licence before use. We update it every January. Corrections: info@uncodemy.com.