UnCodemy

# Deep Learning Mini Project Ideas PDF

A deep learning mini project should be small enough to train on free Google Colab in minutes and teach one idea: a dense network, a CNN, an RNN or transfer learning. Handwritten digit recognition, Fashion-MNIST, cats vs dogs and spam detection with an LSTM are good starting points for 2nd and 3rd year students.

## Key takeaways

- Start with MNIST or Fashion-MNIST to learn the training loop.
- Move to CNNs and then transfer learning in your next project.
- Plot training and validation curves every time; they show overfitting early.
- Keras is the quickest way to start; PyTorch is a good second step.
- Each idea fits in one to four weeks.

## Who this PDF is for

This PDF is for 2nd and 3rd year students who know basic Python and want a first deep learning project for a lab or semester submission. Every idea runs on Google Colab or Kaggle notebooks, uses a well-known public dataset, and builds one skill at a time: dense networks, CNNs, recurrent networks and transfer learning.

## How to choose your project

- **Match your syllabus.** If you have studied CNNs, pick an image project.
- **Keep training short.** Aim for runs under 15 minutes so you can try many settings.
- **Change one thing at a time** and log the result in a table.
- **Write the model yourself** in Keras or PyTorch rather than copying a full notebook.

## Beginner mini projects (1 to 2 weeks)

### 1. Handwritten Digit Recognition

- **Problem:** Recognise digits 0 to 9 from small images.
- **Dataset:** MNIST.
- **Tech stack:** Python, Keras.
- **Main modules:** Data loading, dense network, CNN, drawing-pad demo.
- **Steps:** Train a dense network, then a small CNN, compare accuracy, add a canvas to draw digits.
- **Show in the viva:** Why the CNN beats the dense network.

- **Extensions:** Test on digits you write on paper.

### 2. Fashion Item Classifier

- **Problem:** Classify clothing images into 10 classes.
- **Dataset:** Fashion-MNIST.
- **Tech stack:** Python, Keras or PyTorch.
- **Main modules:** CNN, dropout, batch normalisation, confusion matrix.
- **Steps:** Train a baseline CNN, add dropout and batch norm, compare curves.
- **Show in the viva:** What dropout does to the validation curve.
- **Extensions:** Visualise learned filters.

### 3. Cats vs Dogs with Transfer Learning

- **Problem:** Classify photos as cat or dog.
- **Dataset:** Kaggle "Dogs vs. Cats".
- **Tech stack:** Python, Keras, MobileNetV2.
- **Main modules:** Image pipeline, frozen base model, fine-tuning, prediction script.
- **Steps:** Resize images, train only the top layers, then unfreeze some layers and fine-tune.
- **Show in the viva:** What freezing layers means.
- **Extensions:** Add a third class with your own photos.

### 4. Spam Detection with an LSTM

- **Problem:** Classify SMS messages as spam or ham.
- **Dataset:** UCI "SMS Spam Collection".
- **Tech stack:** Python, Keras.
- **Main modules:** Tokenizer, padding, embedding layer, LSTM, evaluation.
- **Steps:** Tokenise, pad to fixed length, train an LSTM, compare with a Naive Bayes baseline.
- **Show in the viva:** What the embedding layer learns.
- **Extensions:** Swap LSTM for GRU and compare.

## Intermediate mini projects (2 to 3 weeks)

### 5. CIFAR-10 Image Classifier

- **Problem:** Classify small colour images into 10 classes.
- **Dataset:** CIFAR-10.
- **Tech stack:** Python, PyTorch.
- **Main modules:** Data augmentation, CNN, learning-rate schedule, results table.
- **Steps:** Build a CNN, add flips and crops, try a learning-rate schedule, log every run.
- **Show in the viva:** Your results table and which change helped most.
- **Extensions:** Try a small ResNet.

### 6. Sign Language Letter Recognition

- **Problem:** Recognise hand-sign letters from images.
- **Dataset:** Kaggle "Sign Language MNIST".

- **Tech stack:** Python, Keras, OpenCV.
- **Main modules:** CNN classifier, webcam capture, prediction overlay.
- **Steps:** Train a CNN on the grayscale images, crop a hand region from webcam frames, predict live.
- **Show in the viva:** Why webcam accuracy is lower than test accuracy.
- **Extensions:** Spell short words from a sequence of signs.

### 7. Movie Review Sentiment

- **Problem:** Classify movie reviews as positive or negative.
- **Dataset:** IMDB reviews dataset bundled with Keras.
- **Tech stack:** Python, Keras.
- **Main modules:** Embedding, bidirectional LSTM, early stopping.
- **Steps:** Load the built-in IMDB data, train a bidirectional LSTM with early stopping, test on your own sentences.
- **Show in the viva:** Why early stopping helps.
- **Extensions:** Compare with a 1D CNN.

### 8. Autoencoder for Image Denoising

- **Problem:** Remove noise from images.
- **Dataset:** MNIST or Fashion-MNIST with added noise.
- **Tech stack:** Python, Keras.
- **Main modules:** Noise generator, convolutional autoencoder, before/after viewer.
- **Steps:** Add Gaussian noise, train an autoencoder to rebuild clean images, compare outputs.
- **Show in the viva:** What the bottleneck layer does.
- **Extensions:** Use the autoencoder for anomaly detection.

## Stretch mini projects (3 to 4 weeks)

### 9. Stock Price Trend with LSTM (Learning Exercise)

- **Problem:** Predict next-day closing price direction and learn sequence modelling.
- **Dataset:** Public daily price history downloaded with a library such as yfinance.
- **Tech stack:** Python, Keras, pandas.
- **Main modules:** Windowing, LSTM, naive baseline, honest evaluation.
- **Steps:** Build sliding windows, train an LSTM, compare with "tomorrow equals today".
- **Show in the viva:** Why beating the naive baseline is hard, and that this is not trading advice.
- **Extensions:** Add volume and moving-average features.

### 10. Speech Digit Recognition

- **Problem:** Recognise spoken digits.
- **Dataset:** Free Spoken Digit Dataset (FSDD) on GitHub.
- **Tech stack:** Python, librosa, Keras.
- **Main modules:** MFCC features, CNN, microphone test.
- **Steps:** Extract MFCCs, train a small CNN, record and test your own voice.

- **Show in the viva:** What MFCC features capture.
- **Extensions:** Add noise augmentation.

**11. Face Mask Classifier**

- **Problem:** Classify faces as with or without a mask.
- **Dataset:** Kaggle "Face Mask Detection" (andrewmvd).
- **Tech stack:** Python, Keras, OpenCV.
- **Main modules:** Face crop extraction, MobileNetV2 classifier, webcam demo.
- **Steps:** Crop faces from the annotations, fine-tune MobileNetV2, run on webcam frames.
- **Show in the viva:** How you cropped faces from bounding boxes.
- **Extensions:** Add an "incorrectly worn" class.

**12. Neural Style Transfer**

- **Problem:** Repaint a photo in the style of a painting.
- **Dataset:** Your own photos plus public-domain paintings.
- **Tech stack:** Python, TensorFlow, pretrained VGG19.
- **Main modules:** Content and style loss, optimisation loop, gallery page.
- **Steps:** Load VGG19, compute losses, optimise the image, save outputs at several weights.
- **Show in the viva:** How content and style weights change the result.
- **Extensions:** Try fast style transfer models.

## Mini project report structure

A mini project report is usually 10 to 20 pages. Keep it short and clear:

1. **Title page and certificate** in your department's format.
2. **Introduction:** the problem in one paragraph and your objectives as a list.
3. **Tools used:** Keras or PyTorch version, Colab or local machine, GPU type if used.
4. **Design:** one block diagram or flowchart, plus a diagram of your network layers.
5. **Implementation:** what each module does, with 3 to 6 screenshots.
6. **Results:** accuracy or loss curves and a table comparing your runs.
7. **Conclusion and future scope,** then references.

Ask your guide for the exact format before you start writing.

## Common mistakes to avoid

- Training for too many epochs without watching validation loss.
- Forgetting to normalise image pixel values.
- Copying a full Kaggle notebook and not knowing what each layer does.
- Not fixing random seeds, so results change every run.

## Frequently asked questions

**What is the easiest deep learning mini project?**

Handwritten digit recognition on MNIST. It trains in minutes and teaches the full loop of data, model, training and testing.

**Can I do deep learning projects on a laptop without a GPU?**

Small ones like MNIST, yes. For CNNs on colour images, use free Google Colab or Kaggle notebooks.

**Keras or PyTorch for beginners?**

Keras is quicker to start. Learn PyTorch next because many research papers and tutorials use it.

**How do I make my mini project stand out?**

Add a small demo, test it on your own data, and include a table of the experiments you ran.

**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

- Keras code examples
- PyTorch tutorials
- Kaggle Datasets
- Free Spoken Digit Dataset on GitHub

## 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.

More free resources and courses: uncodemy.com | info@uncodemy.com | +91 98183 66550