

Deep Learning Lab - Complete Oral Examination Questions & Answers

Based on BE IT 2019 Lab Manual (Lab Practice-IV)

PRACTICAL 1: Study of Deep Learning Packages

(TensorFlow, Keras, Theano, PyTorch)

Theory-Based Questions:

Q1: What is Deep Learning? A: Deep Learning is a subset of machine learning that uses artificial neural networks with multiple layers (deep networks) to automatically learn and extract features from raw data. It revolutionizes machine learning by showing superior performance compared to traditional ML models for many applications like image recognition, speech processing, and natural language understanding.

Q2: What are the various Python packages for Machine Learning and which are mainly used for Deep Learning? A: Main Python ML packages include:

- **NumPy**: For numerical computations and array operations
- **Pandas**: For data analysis and manipulation
- **Scikit-learn**: For traditional ML algorithms
- **Deep Learning packages**: TensorFlow, Keras, PyTorch, Theano

Q3: Compare TensorFlow, Keras, Theano, and PyTorch (Create a comparison table):

Aspect	TensorFlow	Keras	Theano	PyTorch
Functionality	Complete ML ecosystem	High-level API	Symbolic computing	Flexible dynamic graphs
GUI Status	TensorBoard available	Integrated	Limited	Integrated tools
Version	2.x (TF 2.13+)	3.x	Deprecated (inactive)	2.x (PyTorch 2.0+)
Features	Production-ready, scalable	User-friendly, rapid prototyping	Academic focus	Research-friendly, dynamic
Compatibility	Broad (all platforms)	TensorFlow backend	Python 3.6-3.8	Linux/Windows/Mac
Application Domain	Production, research	Beginners, education	Research (deprecated)	Research, Computer Vision

Q4: List TensorFlow's Models, Datasets, Pre-trained Models, Libraries, Extensions, and Tools: A:

- **Models**: Sequential, Functional, Subclassing
- **Datasets**: MNIST, CIFAR-10, ImageNet

- **Pre-trained Models:** ResNet, VGG, MobileNet
- **Libraries:** TF-Hub, TF-Lite
- **Extensions:** TF-Agents, TF-Probability
- **Tools:** TensorBoard, TensorFlow Lite, TensorFlow.js

Q5: Discuss Two Case Studies using TensorFlow (e.g., PayPal, Intel): A:

1. **PayPal:** Uses TensorFlow for fraud detection in transactions. The system analyzes patterns to identify suspicious activities and protect customers from unauthorized transactions.
2. **Intel:** Uses TensorFlow for optimizing data center efficiency, using deep learning to predict hardware failures and optimize resource allocation.

Q6: Explain the Keras Ecosystem (KerasTuner, KerasNLP, KerasCV, AutoKeras, Model Optimization): A:

- **KerasTuner:** Automated hyperparameter tuning for finding optimal model configurations
- **KerasNLP:** Pre-built models and utilities for natural language processing tasks
- **KerasCV:** Pre-built models and utilities for computer vision applications
- **AutoKeras:** Automated machine learning (AutoML) for automatically designing neural networks
- **Model Optimization:** Techniques to reduce model size and improve inference speed

Q7: Explain Keras Sequential Model Development, Training/Validation, and Parameter Optimization: A:

- **Sequential Model:** Stacks layers one after another linearly
- **Training:** Uses fit() method with training data and validation split
- **Validation:** Automatically splits training data for validation during training
- **Parameter Optimization:** Adjusting learning rate, batch size, epochs, optimizer (Adam, SGD)

Q8: Explain a Simple Theano Program: A: Theano uses symbolic computing. A simple program:

```
import numpy
import theano.tensor as T
from theano import function

x = T.dscalar('x')
y = T.dscalar('y')
z = x + y
f = function([x, y], z)
f(5, 7) # Output: 12
```

Q9: Explain PyTorch Tensors and Applications (Uber's Pyro, Tesla Autopilot): A:

- **PyTorch Tensors:** Multi-dimensional arrays similar to NumPy arrays but optimized for GPU computation
- **Uber's Pyro:** Probabilistic programming library built on PyTorch for Bayesian modeling
- **Tesla Autopilot:** Uses neural networks for autonomous vehicle perception, likely leveraging PyTorch for training

Q10: Why is a Virtual Environment Used for Installing Deep Learning Packages? A: Virtual environments isolate project dependencies, prevent version conflicts, ensure reproducibility, and allow multiple Python projects with different library versions on the same machine.

PRACTICAL 2: Feedforward Neural Networks with Keras & TensorFlow

Theory-Based Questions:

Q1: What is a Feedforward Neural Network (FNN)? A: An FNN is a neural network where data flows in one direction - from input layer through hidden layers to output layer. Each neuron is connected to all neurons in the next layer, with no cycles or loops in the network structure.

Q2: Explain the concept of Tensors in TensorFlow: A: A tensor is a multidimensional array of data. In its simplest form, it's a scalar (0D), then vector (1D), matrix (2D), and higher-order tensors (3D+). All inputs, outputs, and operations in TensorFlow work with tensors.

Q3: What is the difference between Flatten Layer and Dense Layer? A:

- **Flatten Layer:** Converts multi-dimensional input (e.g., 28x28 image) into a 1D vector without changing values
- **Dense Layer:** Fully connected layer that performs weighted sum, adds bias, and applies activation function

Q4: Explain Activation Functions - Why do we need ReLU and Softmax? A:

- **ReLU (Rectified Linear Unit):** Introduces non-linearity; outputs x if $x>0$, else 0. Used in hidden layers for computational efficiency
- **Softmax:** Converts output layer values into probability distribution summing to 1, used for multi-class classification

Q5: What is the difference between categorical_crossentropy and sparse_categorical_crossentropy? A:

- **categorical_crossentropy:** Used when targets are one-hot encoded (e.g., [0,1,0,0])
- **sparse_categorical_crossentropy:** Used when targets are integer labels (e.g., 1 for class 1)

Q6: Explain the SGD Optimizer and its momentum parameter: A:

- **SGD (Stochastic Gradient Descent):** Updates weights based on individual samples instead of entire dataset
- **Momentum:** Accelerates gradient descent by adding a fraction of previous weight update to current update, helping convergence

Q7: What does the fit() method do in Keras? A: The fit() method trains the neural network on training data for specified epochs, computing loss and metrics, updating weights through backpropagation, and optionally using validation data to monitor generalization.

Q8: How do we evaluate a trained model's performance? A: Using the evaluate() method on test data to compute test loss and accuracy. Also use predict() to get predictions

and calculate additional metrics like confusion matrix or precision-recall.

Q9: What information does the training history provide? A: The history object contains dictionaries with:

- Training loss and accuracy for each epoch
- Validation loss and accuracy for each epoch Used to plot graphs and identify overfitting/underfitting

Q10: Why is model accuracy > 90% important for MNIST dataset? A: It indicates the model successfully learned digit patterns. MNIST is relatively easy, so >90% accuracy shows basic network functionality. Below 90% suggests underfitting or poor training.

Q11: What techniques improve model generalization? A:

- **Dropout:** Randomly deactivates neurons during training
- **Batch Normalization:** Normalizes layer inputs
- **Early Stopping:** Stops training when validation loss stops improving
- **L1/L2 Regularization:** Penalizes large weights

Q12: Explain the relationship between batch size, epochs, and learning rate: A:

- **Batch Size:** Number of samples processed before weight update (larger = faster but less accurate)
- **Epochs:** Complete passes through training data (more epochs = more learning but risk of overfitting)
- **Learning Rate:** Controls how much weights change per update (high = fast but unstable, low = stable but slow)

PRACTICAL 3: Image Classification using CNN

Theory-Based Questions:

Q1: What is a Convolutional Neural Network (CNN)? A: CNN is a specialized neural network designed for processing grid-like data (images). It automatically learns spatial hierarchies of features through convolutional layers, making it highly effective for image recognition and classification tasks.

Q2: What are the three main groups in CNN architecture? A:

1. **Input Layer:** Accepts image data with dimensions (width × height × channels)
2. **Feature Extraction Layers:** Convolutional layers extract features; pooling layers downsample
3. **Classification Layers:** Fully connected layers produce class probabilities

Q3: Explain the Convolutional Layer: A: The convolutional layer applies learnable filters (kernels) across the image, computing dot products between filter and input patches. Each filter learns to detect specific features (edges, textures, shapes). Multiple filters create multiple feature maps.

Q4: What is the difference between traditional Neural Networks and CNNs? A:

- **Traditional NN:** Every neuron connects to all neurons in next layer (fully connected), treats images as flat vectors
- **CNN:** Neurons connect only to local regions (local connectivity), preserves spatial information, uses shared weights (fewer parameters)

Q5: Explain Pooling Operation: A: Pooling downsamples feature maps by selecting maximum (max pooling) or average (average pooling) value from small patches. This reduces spatial dimensions, decreases computation, and provides translation invariance.

Q6: What is Max Pooling and why is it used? A: Max pooling selects the maximum value from a window (e.g., 2x2). It's used to:

- Reduce spatial dimensions
- Decrease parameters and computation
- Provide translation invariance
- Highlight strongest features

Q7: Explain the Convolution Operation with an example: A: Given array $a=[2, 5, 8, 4, 7, 9]$ and filter $b=[1, 2, 3]$:

- Element-wise multiply: $[2 \times 1, 5 \times 2, 8 \times 3] = [2, 10, 24]$
- Sum: $2+10+24 = 36$ (first output value)
- Slide filter and repeat for all positions

Q8: Why do we normalize pixel values to [0,1] range? A: Neural networks converge faster with normalized inputs. Original pixel values [0, 255] have large scale variation. Normalizing to [0,1] improves training stability and convergence speed.

Q9: Explain the Flatten Layer: A: The Flatten layer converts multi-dimensional feature maps (e.g., 13x13x32) into a 1D vector (5408,). This converts spatial information into a format suitable for fully connected layers that produce class scores.

Q10: What does "64 filters of size 3x3" mean in Conv2D(64, (3,3))? A:

- **64 filters:** 64 learnable convolutional kernels, each detecting different features
- **3x3 size:** Each filter is a 3x3 matrix applied across the image
- **Output:** 64 different feature maps, each showing detected features

Q11: Why is test accuracy 98.55% better than training accuracy sometimes? A: Typically training accuracy > test accuracy due to overfitting. Higher test accuracy is unusual but may occur due to regularization effects, lucky test set, or model generalization working well.

Q12: Explain feature extraction in CNNs: A: Lower layers learn low-level features (edges, textures), middle layers combine them into mid-level features (shapes), higher layers learn high-level features (objects, patterns). This hierarchical feature learning is automatic and powerful.

Q13: Why are we reshaping images to (28,28,1) instead of (28,28)? A: CNN layers expect 4D tensors: (batch_size, height, width, channels). The extra dimension (1) represents the color channel. For grayscale images, it's 1; for RGB, it's 3.

Q14: How does image classification differ from traditional classification? A: Image classification uses CNNs to automatically learn spatial features, while traditional classification uses hand-crafted features. CNNs are more accurate and scalable for image data.

PRACTICAL 4: Anomaly Detection using Autoencoder

Theory-Based Questions:

Q1: What is an Autoencoder? A: An autoencoder is an unsupervised neural network that learns to compress data into a latent representation (encoding) and reconstruct it (decoding). It consists of an encoder that reduces dimensionality and decoder that restores original dimensions.

Q2: What is the difference between Encoder and Decoder? A:

- **Encoder:** Compresses input data into compact latent representation with lower dimensions
- **Decoder:** Reconstructs original input from latent representation
- Together, they form an autoencoder where input = output ideally

Q3: What is latent representation? A: The compressed representation in the middle bottleneck layer containing essential information from input. It has lower dimensions than input, forcing the network to learn most important features.

Q4: How does Autoencoder work for Anomaly Detection? A:

1. Train autoencoder on normal data only
2. Calculate reconstruction error (difference between input and reconstructed output)
3. Normal data has low reconstruction error
4. Anomalies have high reconstruction error (model hasn't seen them)
5. Set threshold above which data points are classified as anomalies

Q5: Why train Autoencoder only on normal data? A: If trained on mixed data (normal + anomalies), the network learns to reconstruct anomalies too, defeating the purpose. Training only on normal data ensures low reconstruction error only for normal patterns.

Q6: What is reconstruction loss and why is Mean Squared Error (MSE) used? A:

Reconstruction loss measures difference between input and output. MSE is used because:

- Provides smooth gradient for backpropagation
- Penalizes large errors more than small ones
- Works well for continuous data

Q7: What does a threshold value of 52 mean in anomaly detection? A: Any data point with reconstruction error > 52 is classified as anomaly. Threshold is determined by analyzing reconstruction error distribution and choosing value that separates normal from anomalous data effectively.

Q8: Why is the dataset highly imbalanced in anomaly detection? A: Real-world datasets contain far more normal data than anomalies (e.g., 99.5% normal, 0.5% anomalies). This imbalance makes accuracy a poor metric; precision and recall are more informative.

Q9: Explain the Autoencoder architecture (Encoder → Latent → Decoder): A:

- **Encoder:** Multiple dense layers progressively reducing dimensions
- **Latent Layer:** Bottleneck with smallest dimensions (e.g., 32 units)
- **Decoder:** Mirror of encoder, progressively increasing dimensions back to input size

Q10: What improvements can be made to increase recall and precision? A:

- Add more relevant features

- Use different autoencoder architecture
- Adjust hyperparameters (learning rate, batch size, epochs)
- Use different algorithms like Isolation Forest or One-Class SVM

Q11: Why normalize/scale data before training Autoencoder? A: Scaling ensures all features contribute equally to reconstruction error regardless of their original scale. Without scaling, features with larger values would dominate the loss.

Q12: What optimizer and loss function are used for Autoencoder? A:

- **Optimizer:** Adam (adaptive learning rate, works well for most cases)
 - **Loss Function:** Mean Squared Error (MSE) for continuous data
 - **Alternative:** Binary Crossentropy for binary reconstruction
-

PRACTICAL 5: Continuous Bag of Words (CBOW) Model

Theory-Based Questions:

Q1: What is Natural Language Processing (NLP)? A: NLP is the field of AI that focuses on enabling computers to understand, interpret, and generate human language. It involves tasks like text classification, machine translation, sentiment analysis, and named entity recognition.

Q2: What is Word Embedding? A: Word embedding represents words as dense vectors of real numbers in a continuous vector space. Words with similar meanings have similar embeddings. It captures semantic and syntactic relationships between words.

Q3: Explain Word2Vec Techniques: A: Word2Vec includes two approaches:

- **CBOW (Continuous Bag of Words):** Predicts target word from context words
- **Skip-gram:** Predicts context words from target word Both use neural networks to learn word embeddings efficiently

Q4: What are applications of Word Embedding in NLP? A:

- Sentiment analysis
- Machine translation
- Text classification
- Named entity recognition
- Question answering systems
- Semantic similarity

Q5: Explain CBOW Architecture: A: CBOW has three layers:

- **Input Layer:** Context words represented as one-hot encoded vectors
- **Hidden Layer (Embedding):** Projects input to lower-dimensional space (learns embeddings)
- **Output Layer:** Predicts target word using softmax activation

Q6: What is input and output to CBOW model? A:

- **Input:** Context words (e.g., "quick", "fox", "jumps" for target word "brown")
- **Output:** Probability distribution over vocabulary, predicting target word "brown"

Q7: What is Tokenizer? A: Tokenizer converts raw text into tokens (individual words) and assigns unique indices to each word. It creates vocabulary and enables word-to-

index and index-to-word mappings.

Q8: Explain window size parameter in detail for CBOW: A: Window size determines how many context words surround the target word:

- Window size = 2: Takes 2 words before and 2 words after target word (total 4 context words)
- Larger window: Captures broader context but more computation
- Smaller window: Captures local context, faster training

Q9: Explain Embedding and Lambda Layer from Keras: A:

- **Embedding Layer:** Maps integer word indices to dense vectors of embeddings. Learns these embeddings during training
- **Lambda Layer:** Custom layer applying arbitrary function to data. Used to average embeddings or perform custom operations

Q10: What is yield() function? A: `yield()` is a generator function that produces data one batch at a time instead of loading all data at once. Useful for large datasets to manage memory efficiently during training.

Q11: What does padding sequences mean? A: Padding adds zeros to shorter sequences to make all sequences equal length. Necessary because neural networks expect fixed-size inputs. Handles variable-length context windows.

Q12: Explain one-hot encoding: A: Represents categorical variable as binary vector with one element as 1 and rest as 0. For word "cat" (index 3 in vocab), one-hot encoding might be `[0,0,0,1,0,...]`.

Q13: How does CBOW learn word embeddings? A: Through backpropagation:

1. Forward pass: Context words → Hidden (embeddings) → Output (prediction)
2. Calculate loss between predicted and actual target word
3. Backpropagate error
4. Update embedding weights
5. Repeat for many examples

Q14: What does "most_similar()" function do? A: Finds words with embeddings most similar to query word in vector space using cosine similarity. Example: `most_similar(positive=['king'])` returns words semantically similar to "king".

PRACTICAL 6: Transfer Learning with CNN

Theory-Based Questions:

Q1: What is Transfer Learning? A: Transfer learning reuses knowledge learned from one task/dataset to solve another similar task. Instead of training from scratch, start with pre-trained weights, saving time and improving accuracy, especially with limited data.

Q2: What are Pre-trained Neural Network Models? A: Models trained on large datasets (e.g., ImageNet with 1.2M images, 1000 classes) by large organizations. Weights capture general visual features. Examples: VGG-16, ResNet, MobileNet, Inception.

Q3: Explain PyTorch library in short: A: PyTorch is a deep learning framework offering:

- Dynamic computational graphs (define-by-run)
- GPU acceleration
- Pythonic interface
- Strong community support
- Useful for research and production

Q4: What are advantages of Transfer Learning? A:

- **Speed:** Faster training (start from pre-trained weights)
- **Accuracy:** Better performance with limited data
- **Efficiency:** Reduced computational resources
- **Generalization:** Pre-trained features often transfer well
- **Cost:** Lower training cost

Q5: What are applications of Transfer Learning? A:

- Medical image analysis (limited data)
- Object detection
- Face recognition
- Sentiment analysis
- Speech recognition
- Video classification

Q6: Explain Caltech 101 Images Dataset: A: Dataset containing 101 object categories with 40-800 images per category (total ~9,144 images). Used for object recognition and transfer learning research. Images are relatively small with variable sizes.

Q7: Explain ImageNet Dataset: A: Large-scale visual database with 1.2M training images across 1000 object categories. Images are labeled and hierarchically organized. De facto standard for pre-training computer vision models.

Q8: List basic steps for Transfer Learning: A:

1. Load pre-trained model (trained on large dataset)
2. Freeze lower layers (keep learned features)
3. Replace/modify top layers for new task
4. Train only top layers (fine-tuning)
5. Optionally unfreeze lower layers and train entire network

Q9: What is Data Augmentation? A: Artificially increasing training data by applying random transformations:

- Random crops
- Rotations
- Flipping
- Color jittering
- Resizing Improves model robustness and generalization

Q10: How and why is Data Augmentation done in Transfer Learning? A:

- **How:** Apply transformations during training to create variations of images
- **Why:**
 - Increases effective dataset size
 - Reduces overfitting
 - Improves robustness to variations
 - Particularly important when fine-tuning dataset is small

Q11: Why is preprocessing needed on input data in Transfer Learning? A:

- Pre-trained models expect specific input format (e.g., 224x224 for VGG)
- Trained on normalized images with specific mean/std values
- Must apply identical preprocessing: resize, normalize with same statistics
- Ensures model receives data in expected format

Q12: What is PyTorch Transforms Module? A: Collections of image transformation functions. Example transforms:

- RandomResizedCrop : Crop and resize to random size
- RandomRotation : Rotate image by random angle
- ColorJitter : Randomly change brightness/contrast/saturation
- RandomHorizontalFlip : Flip image horizontally randomly
- CenterCrop : Crop from center
- ToTensor : Convert PIL image to PyTorch tensor
- Normalize : Normalize with specific mean/std (ImageNet: mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])

Q13: Explain Validation Transforms steps: A: Validation transforms differ from training (no random augmentation):

- Resize(256) : Resize to 256x256
- CenterCrop(224) : Crop center 224x224 (ImageNet standard)
- ToTensor : Convert to tensor
- Normalize : Apply ImageNet normalization No random augmentation ensures consistent evaluation

Q14: Explain VGG-16 Model: A:

- **Architecture:** 16 layers (13 convolutional + 3 fully connected)
- **Features:** Simple, stacked 3x3 convolutional filters, max pooling
- **Input:** 224x224 RGB image
- **Layers:** 5 blocks of convolutional layers + 3 dense layers
- **Parameters:** 138M (large model)
- **Pre-training:** Trained on ImageNet
- **Use:** Base for transfer learning, good for small datasets

Q15: How to freeze model parameters? A:

```
for param in model.parameters():
    param.requires_grad = False
```

This prevents gradients from being computed and weights from being updated during training, preserving pre-trained knowledge

Q16: What happens when we add a custom classifier? A: Replace the original classification layer with new layers:

- New fully connected layers with architecture matching new problem
- Output size = number of new classes
- Only these layers are trained (other layers frozen)
- Transfers learned features to new task

Q17: Explain early stopping concept: A: Stop training when validation loss stops improving to prevent overfitting:

- Monitor validation loss/accuracy each epoch
- If no improvement for N epochs, stop training

- Save best model weights
- Prevents wasting computation and improves generalization

Q18: What is unfreezing layers? A: After training top layers, unfreeze some lower layers and train entire network with very low learning rate. Allows fine-tuning of pre-trained features for specific task while preserving general knowledge.

Quick Reference: Common Deep Learning Concepts

Q: What is Overfitting? A: Model learns training data too well including noise, performs poorly on new data. Indicated by large gap between training and validation accuracy.

Q: What is Underfitting? A: Model is too simple to capture data patterns, performs poorly on both training and validation data.

Q: What is Backpropagation? A: Algorithm computing gradients of loss with respect to weights, enabling efficient weight updates in multiple layers.

Q: What is Gradient Descent? A: Optimization algorithm updating weights in direction of negative gradient to minimize loss function.

Q: What is Batch Normalization? A: Normalizes layer inputs to have zero mean and unit variance, improving training stability and speed.

Q: What is Dropout? A: Regularization technique randomly deactivating neurons during training to prevent co-adaptation and reduce overfitting.

Q: What is One-Hot Encoding? A: Represents categorical variable as binary vector with single 1 and rest 0s.

Q: What is Cross-Entropy Loss? A: Measures difference between predicted probability distribution and actual labels, standard for classification.

Q: What does Epoch mean? A: One complete pass through entire training dataset.

Q: What is Batch Size? A: Number of samples processed before updating weights, trade-off between speed and accuracy.

How to Use This Guide for Exam Preparation

1. **Read all Q&A pairs thoroughly** - Understand not just answers but why they're correct
 2. **Connect concepts** - Link topics across practicals (e.g., CNN uses similar optimization as FNN)
 3. **Refer to manual** - Use provided theory sections to deepen understanding
 4. **Practice explaining** - Try explaining concepts without reading to test understanding
 5. **Ask follow-up questions** - For each answer, think "why?" and "how?"
 6. **Study diagrams** - Review figures in manual alongside conceptual explanations
 7. **Code understanding** - Understand code snippets and why each line is necessary
-

Important Points to Remember

- Always mention practical applications when discussing concepts
 - Understand differences between related concepts (e.g., FNN vs CNN)
 - Be familiar with all mentioned datasets (MNIST, CIFAR-10, ImageNet, Caltech-101)
 - Know installation steps and why virtual environments are used
 - Understand activation functions and when to use each one
 - Remember key hyperparameters and their effects
 - Be prepared to explain your project implementation
 - Know advantages and disadvantages of each framework
 - Understand loss functions and optimization algorithms
 - Be ready for follow-up questions about code implementation
-

This comprehensive guide covers all six practicals with deep theory and practical understanding. Best of luck with your viva examination!