Deep Learning Cheat Sheet

by Naresh Edagotti

"Every neuron you understand is a step closer to building intelligence."

What is ANN? | Core Components | Case Study | Learn-by-Doing

Section 1: What is a Neural Network (ANN)?

Understanding the foundation of artificial neural networks

Definition in Simple Words

A Neural Network is a computer system inspired by the human brain that learns from examples. Just like your brain learns to recognize faces by seeing many faces, a neural network learns to recognize patterns by seeing many examples.

Real-world Analogy: The Brain's Neurons

Brain Neurons

- Neurons receive signals through dendrites
- Process signals in the cell body
- Transmit signals through axons
- Connect to other neurons via synapses

Artificial Neurons

- Receive inputs through connections
- Process inputs with weights and biases
- Apply activation function
- Pass output to next layer

How it Works: Input → Hidden Layers → Output

Input Layer → Hidden Layers → Output Layer

Input Layer

Receives raw data (images, text, numbers, etc.)

Hidden Layers

Process and transform data through mathematical operations

Output Layer

Produces the final prediction or classification

Deep Learning Cheat Sheet

by Naresh Edagotti

"Every neuron you understand is a step closer to building intelligence."

Section 2: Core Components of Neural Networks

Understanding the building blocks of deep learning

Weights & Biases

What are weights and biases?

Weights: Connection strengths between neurons. Higher weights mean stronger influence.

Biases: Threshold values that shift the activation function, making it easier or harder for neurons to fire.

How they affect learning

During training, the network adjusts weights and biases to minimize prediction errors.

Example: Predicting house prices
• Weight for "square footage": +0.8 (positive correlation)
• Weight for "distance to city": -0.3 (negative correlation)
• Bias: +50,000 (base price adjustment)

Activation Functions

Activation functions introduce non-linearity, allowing neural networks to learn complex patterns.

ReLU

$f(x) = \max(0, x)$

Use in hidden layers. Fast computation, avoids vanishing gradient.

Sigmoid

$f(x) = 1/(1 + e^{-x})$

Use in output layer for binary classification (0 to 1).

Tanh

$f(x) = (e^{x} - e^{-x})/(e^{x} + e^{-x})$

Softmax

$f(x_i) = e^{x_i} / \Sigma e^{x_j}$

Use in hidden layers. Outputs between -1 and 1.

Use in output layer for multi-class classification.

Python Code Example

```
import torch
import torch.nn as nn

# Different activation functions
relu = nn.ReLU()
sigmoid = nn.Sigmoid()
tanh = nn.Tanh()
softmax = nn.Softmax(dim=1)

# Example usage
x = torch.tensor([-1.0, 0.0, 1.0])
print("ReLU:", relu(x))
print("Sigmoid:", sigmoid(x))
print("Tanh:", tanh(x))
```

✅ Forward Propagation

Forward propagation is the process of passing input data through the network to get predictions.

Step-by-step calculation

1. Input data enters the network
2. Each neuron calculates: output = activation(weights × inputs + bias)
3. Results pass to the next layer
4. Final layer produces predictions

Simple Math Example (2 neurons)

```
# Input: x1 = 2, x2 = 3
# Weights: w1 = 0.5, w2 = 0.8
# Bias: b = 1

# Calculation:
z = (w1 * x1) + (w2 * x2) + b
z = (0.5 * 2) + (0.8 * 3) + 1
z = 1 + 2.4 + 1 = 4.4

# Apply ReLU activation:
output = max(0, z) = max(0, 4.4) = 4.4
```

✅ Loss Functions

Loss functions measure how well the model is performing by calculating the difference between predictions and actual values.