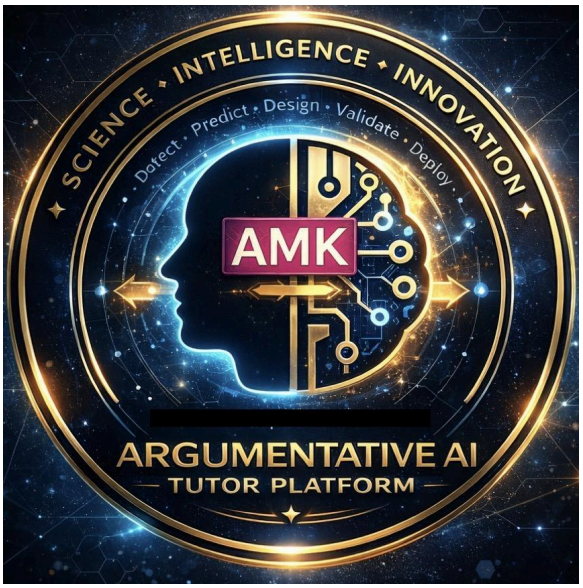


Benchmark Lecture Note 15
Deep Learning and Neural Networks

AI Certification Program • AMK Research Lab Program
Focus: multilayer perceptrons, forward propagation, backpropagation, gradient descent, activation functions, training risks, and responsible model development.

What learners master	Why it matters	AMK governance lens
Neural network basics, loss functions, gradients, and weight updates	Backpropagation powers modern vision, language, tabular, and multimodal learning	Accuracy, explainability, training discipline, monitoring, and human oversight

Learning outcomes

- Explain how neural networks transform inputs through layers, activations, and learned weights.
- Differentiate forward propagation from backpropagation and connect both to gradient descent.
- Interpret the role of activation functions, loss functions, learning rate, and initialization.
- Recognize vanishing and exploding gradients and identify mitigation approaches.
- Use basic Python code to train a small neural network and inspect model outputs responsibly.
- Relate model training to AMK governance themes such as traceability, explainability, safety, and oversight.

1. Deep learning in context

Deep learning uses layered function approximation to learn patterns directly from data. In a feedforward neural network, each layer applies a linear transformation and a nonlinear activation; training then adjusts weights to reduce a loss function through backpropagation and gradient-based optimization. Modern

frameworks automate gradient computation using reverse-mode automatic differentiation, which is the software mechanism behind practical backpropagation.

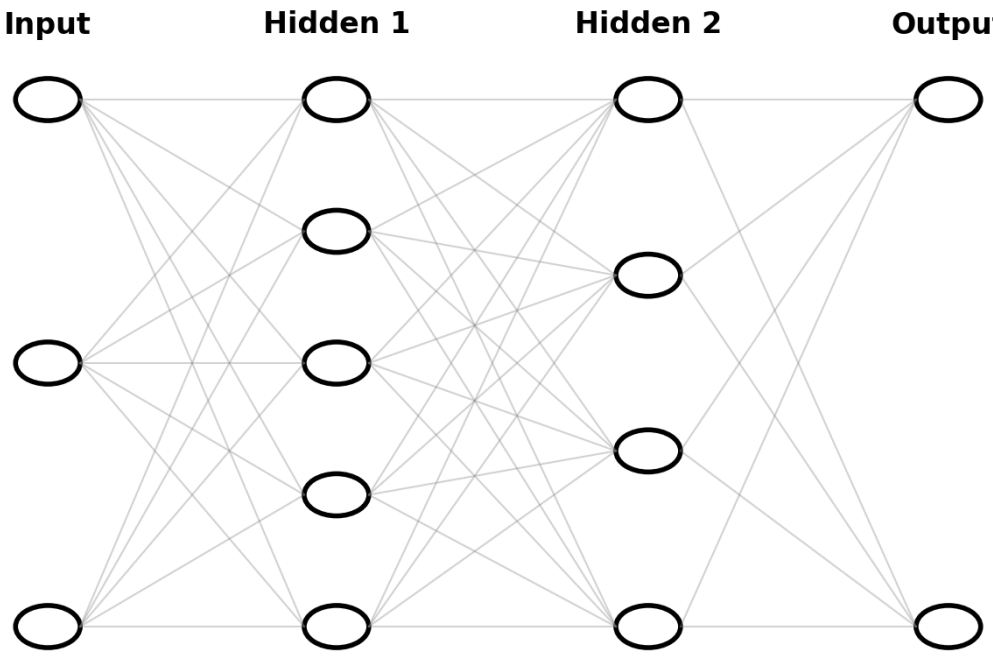


Figure 1. Simple feedforward network with two hidden layers. In forward propagation, information moves left to right. In backpropagation, gradients move right to left to update parameters.

2. Core concepts and vocabulary

Concept	Plain meaning	Mathematical idea	Why it matters
Neuron	A small computing unit	$z = Wx + b$, $a = \varphi(z)$	Combines inputs and nonlinear decision-making
Layer	A collection of neurons	$h(l)=\varphi(W(l)h(l-1)+b(l))$	Builds hierarchical representations
Loss	Measures prediction error	$L(y,\hat{y})$	Objective minimized during training
Gradient	Sensitivity of loss to a parameter	$\partial L/\partial w$	Tells the optimizer which direction to move
Learning rate	Step size for updates	$w \leftarrow w - \eta \partial L/\partial w$	Controls speed and stability of learning

3. Forward propagation and backpropagation

Forward propagation: the network computes hidden states and an output prediction from the input. **Backpropagation:** the training system applies the chain rule backward through the computational graph to compute gradients of the loss with respect to each weight and bias. Those gradients are then used by optimizers such as stochastic gradient descent or Adam to update parameters.

Forward pass	Backward pass
1. Read input features.	1. Differentiate the loss at the output layer.
2. Compute weighted sums and activations layer by layer.	2. Propagate derivatives backward using the chain rule.
3. Produce logits or probabilities.	3. Accumulate gradients for weights and biases.
4. Compare predictions with targets using a loss function.	4. Update parameters with an optimizer.

4. Activation functions and gradient behavior

Activation functions inject nonlinearity so the network can learn complex decision boundaries. Their derivatives strongly influence gradient flow and training stability.

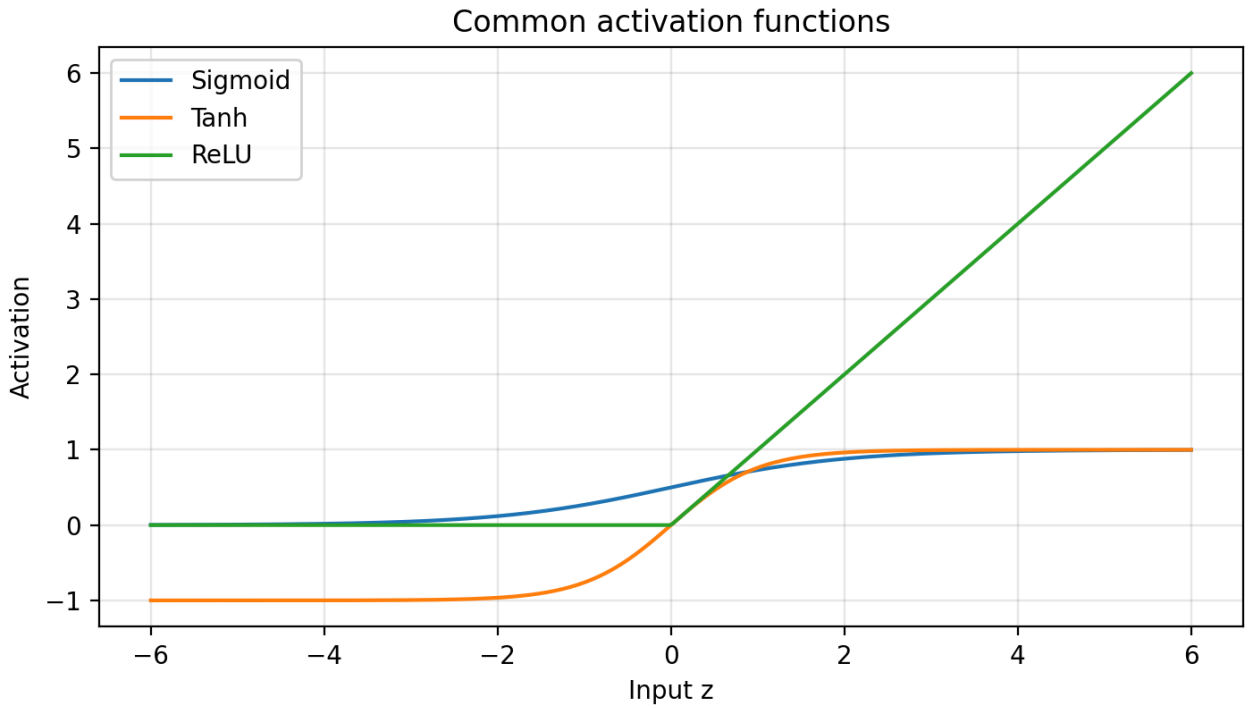


Figure 2. Sigmoid, tanh, and ReLU are common nonlinearities used in hidden or output layers depending on task design.

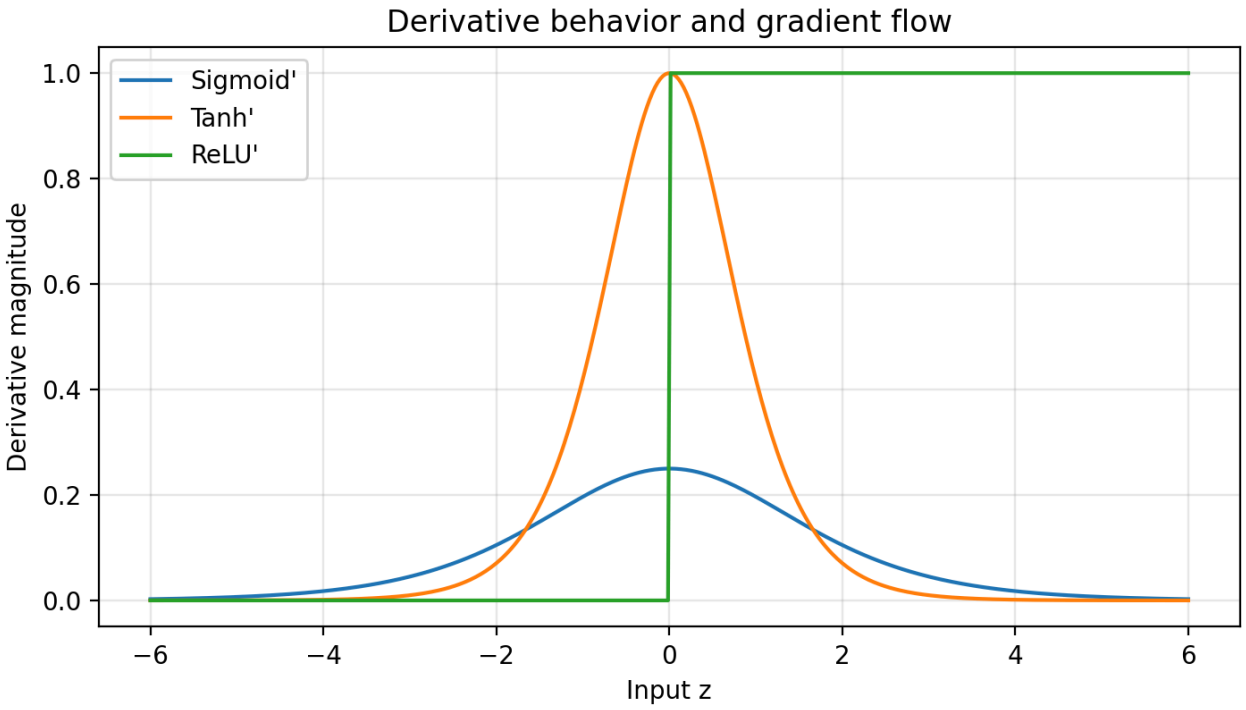


Figure 3. Derivative behavior matters: sigmoid and tanh can saturate, while ReLU keeps a constant derivative for positive inputs.

5. Vanishing and exploding gradients

When many derivatives are multiplied across depth, gradients can shrink toward zero or grow uncontrollably. Vanishing gradients slow learning in earlier layers, while exploding gradients destabilize optimization.

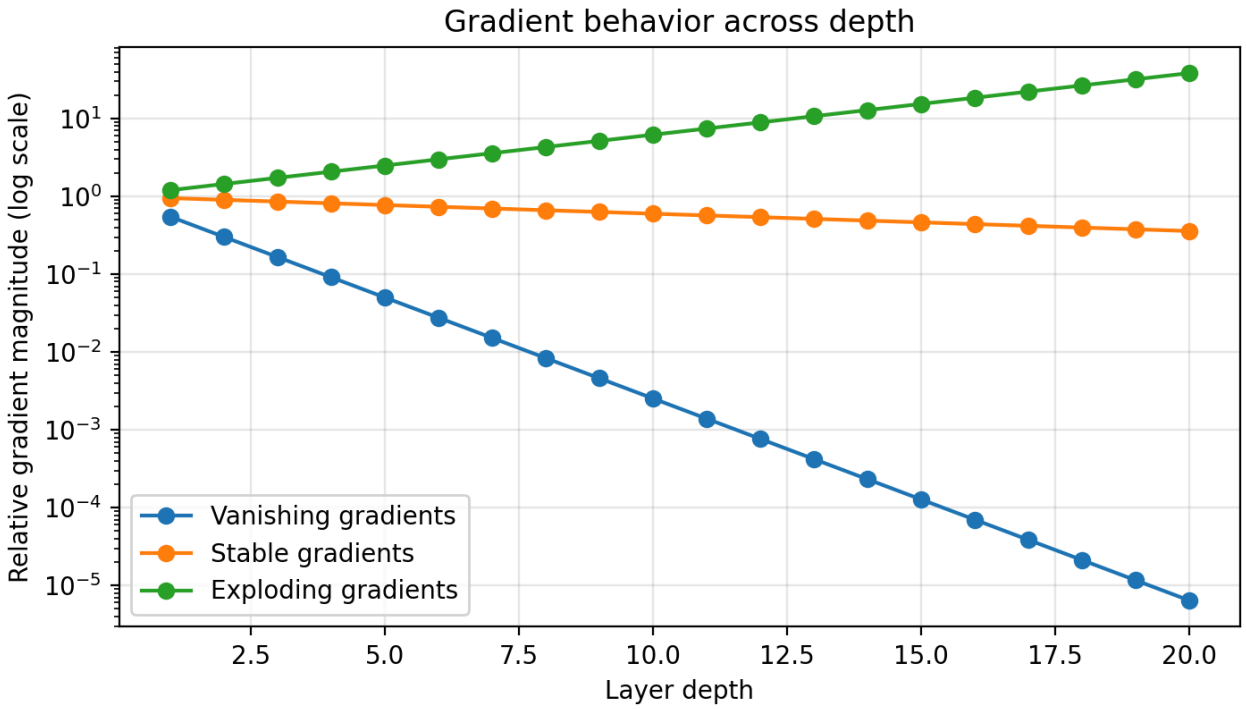


Figure 4. Relative gradient magnitude often decays or blows up across layer depth unless architecture and optimization choices are well controlled.

Problem	Symptoms	Mitigation
Vanishing gradients	Slow early-layer learning; accuracy plateaus	ReLU-family activations, normalization, residual connections, better initialization
Exploding gradients	Loss spikes; NaNs; unstable updates	Gradient clipping, smaller learning rate, normalization, careful initialization
Overfitting	Training improves but validation worsens	Regularization, dropout, data augmentation, early stopping

6. Typical training curve

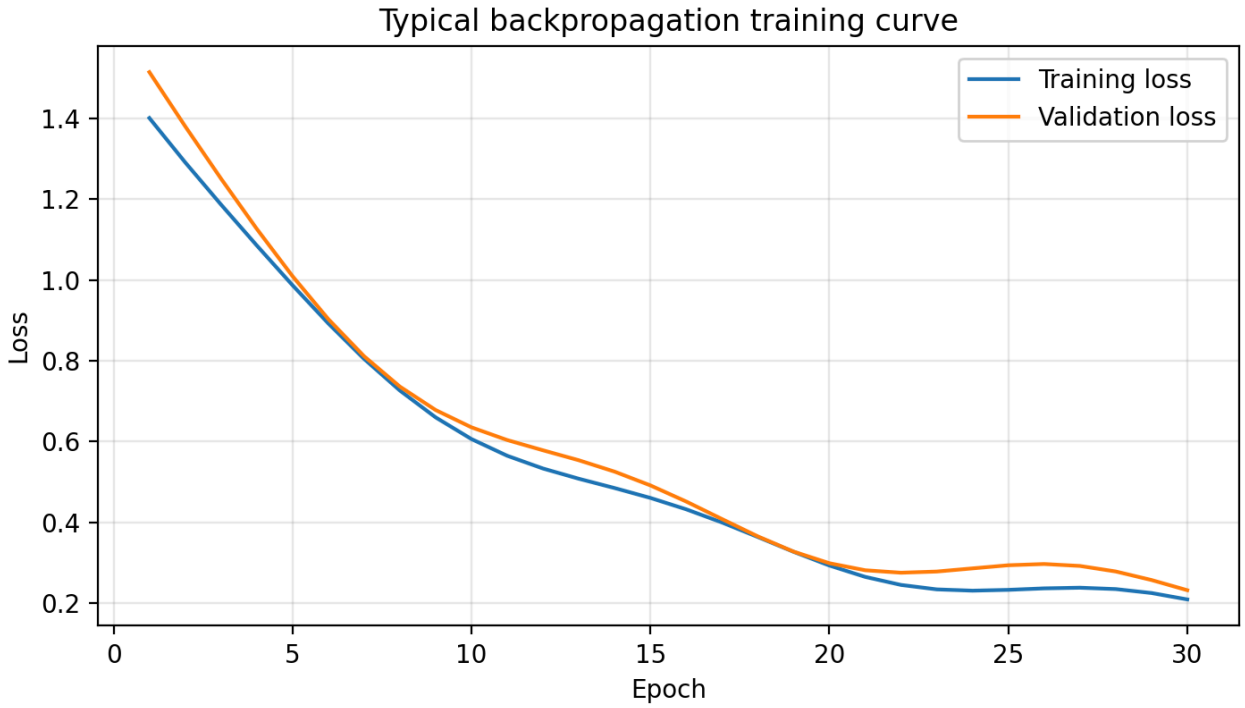


Figure 5. Training and validation loss usually fall together early; later divergence can indicate overfitting or data mismatch.

7. Sample Python code: tiny neural network with manual-style training flow

```
import torch
import torch.nn as nn
import torch.optim as optim
```

```
X = torch.tensor([[0.,0.],[0.,1.],[1.,0.],[1.,1.]])
y = torch.tensor([[0.],[1.],[1.],[0.]])

model = nn.Sequential(
    nn.Linear(2, 8),
    nn.ReLU(),
    nn.Linear(8, 1),
    nn.Sigmoid()
)

criterion = nn.BCELoss()
optimizer = optim.Adam(model.parameters(), lr=0.05)

for epoch in range(2000):
    optimizer.zero_grad()          # clear old gradients
    y_hat = model(X)               # forward pass
    loss = criterion(y_hat, y)     # compute loss
    loss.backward()                # backpropagation
    optimizer.step()               # update parameters

with torch.no_grad():
    preds = model(X)
    print(preds.round())
```

8. AMK governance and responsible training

The AMK Research Lab framing connects technical training with governance-by-design. Deep learning systems should not be judged only by accuracy; they should also be evaluated for traceability, robustness, explainability, and safe deployment.

Training discipline	Operational control	Responsible deployment
Version data splits	Log hyperparameters	Monitor drift and performance
Review loss curves	Track seeds and checkpoints	Use human escalation for high-stakes outputs
Test edge cases	Inspect gradients and failures	Document limitations and known risks

9. AMK research-lab alignment

The AMK S²I and HACE visuals frame deep learning within science, intelligence, innovation, and human-authority safety. In practice, backpropagation belongs to the science and intelligence layers, while deployment guardrails, oversight, and escalation belong to the innovation and safety envelope.

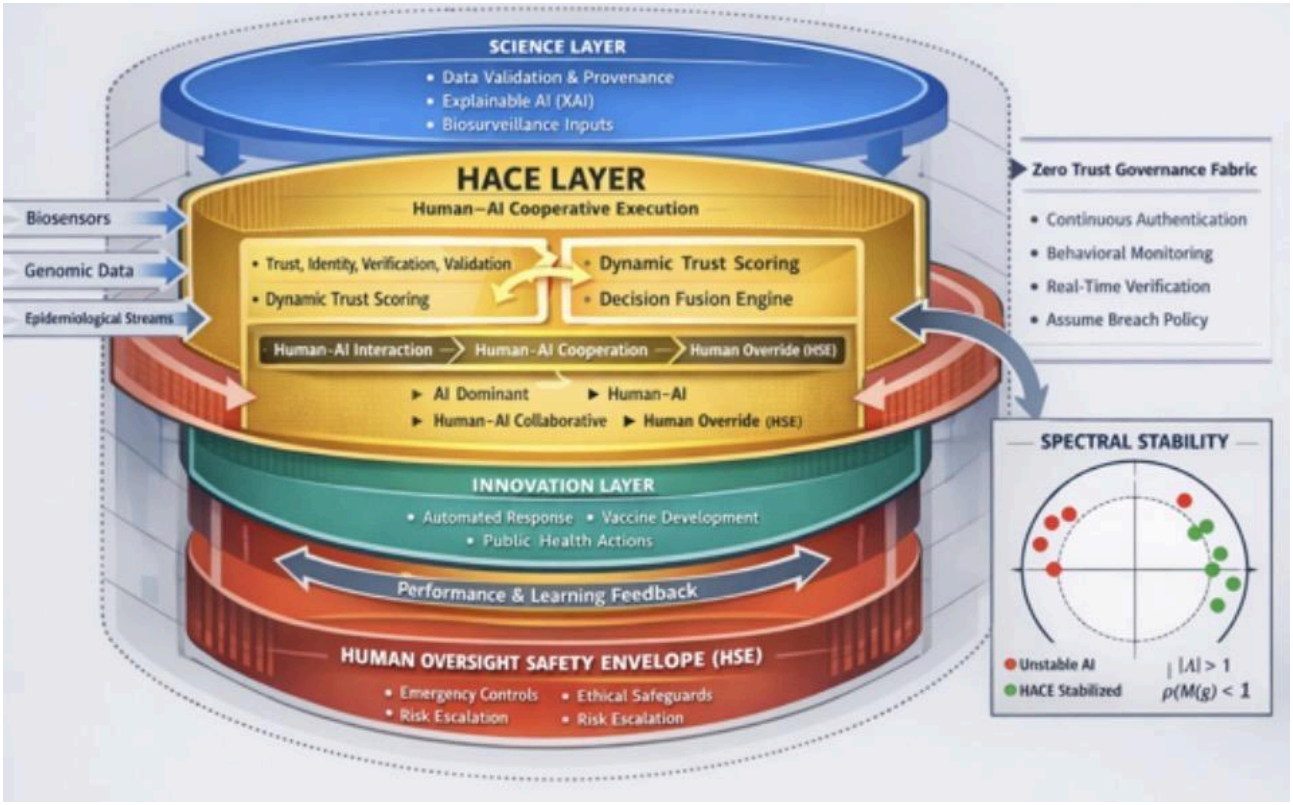
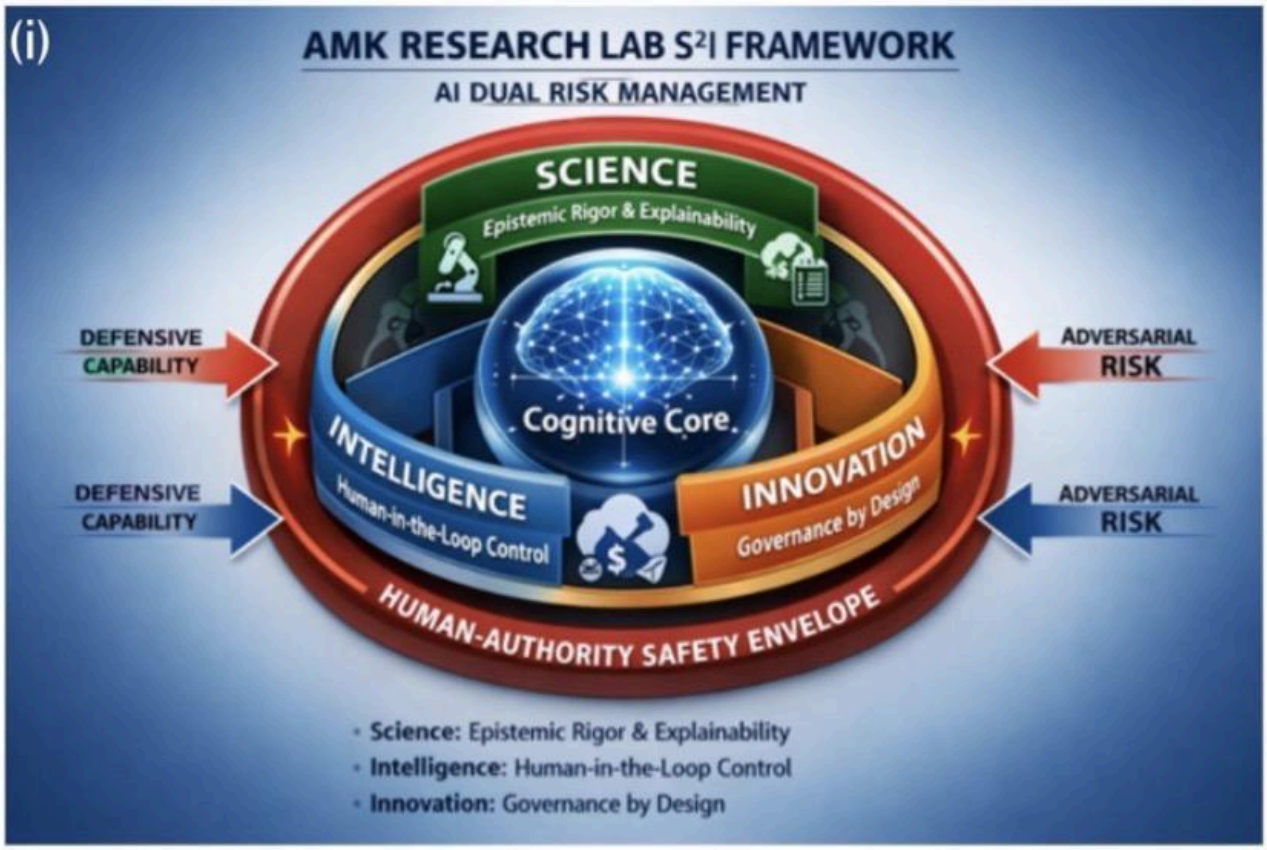


Figure 6. HACE-oriented governance can be used to decide when model outputs remain autonomous and when human override is required.



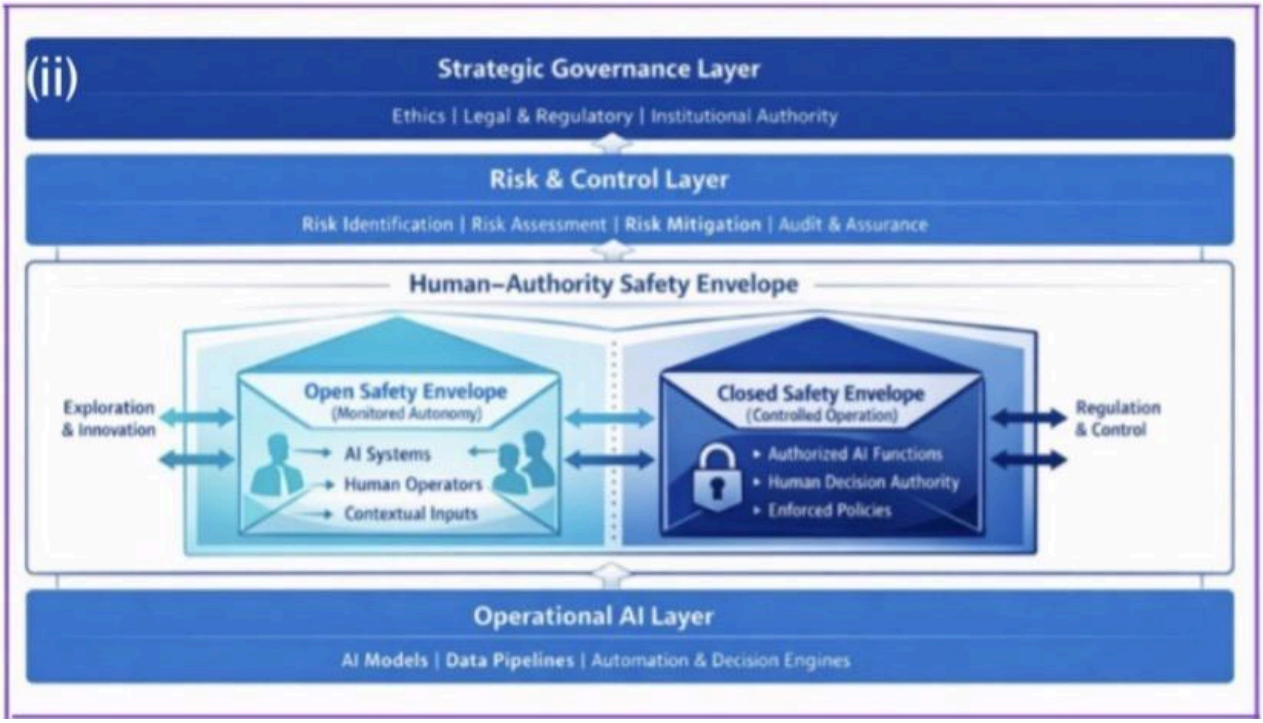


Figure 7. The S²I framework links epistemic rigor, human-in-the-loop control, and governance-by-design for AI systems.

10. Suggested lab activities

1. Train a tiny multilayer perceptron on a binary classification dataset and plot its loss curve.
2. Swap sigmoid, tanh, and ReLU in the hidden layer and compare convergence speed.
3. Increase the learning rate until optimization becomes unstable; then add gradient clipping.
4. Write a short report explaining where the model is accurate, where it fails, and what governance controls should be added before deployment.

11. Capstone directions

Backpropagation is a foundation topic, so Benchmark Lecture Note 15 can feed directly into capstones such as an AI fraud detection model, an intrusion detection system, a healthcare prediction model, or an argumentative tutor that shows learners how gradients and updates affect predictions.

References

- Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. Chapter 6.
- National Institute of Standards and Technology. (2023). AI Risk Management Framework (AI RMF 1.0).
- National Institute of Standards and Technology. (2024). Generative AI Profile.
- PyTorch Tutorials. A Gentle Introduction to torch.autograd.
- PyTorch Documentation. Autograd mechanics.
- TensorFlow. Introduction to gradients and automatic differentiation.
- Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. Dive into Deep Learning. Sections on forward and backward propagation.