Deep Learning for Sequences

AI × Biotechnology — Student Learning Guide

WEEKS 7–9 · APPLIED · INTRO COLLEGE

■ LEARNING OUTCOMES

- Understand: Neural Networks from Scratch
- Understand: 1D CNNs for DNA
- Understand: Transformers & Self-Attention
- Understand: ESM-2 Protein Language Model

■ Duration	■ Level	■ Language	■ Track
Weeks 7–9	Applied	Python 3.10+	Intro College

■ ESM-2 trained on 250M protein sequences predicts secondary structure with 80%+ accuracy — without ever seeing a 3D structure during training.

■ Overview — Deep Learning for Sequences

Deep learning has transformed bioinformatics. Convolutional neural networks detect motifs in DNA. Transformers trained on millions of protein sequences understand amino acid 'language' well enough to predict structure. This week you build these models from scratch in PyTorch.

01. Neural Networks from Scratch

A neural network is a directed graph of mathematical operations. Each layer applies a learned linear transformation followed by a nonlinearity. Training adjusts all weights simultaneously via backpropagation — the chain rule applied recursively.

Concept	Details
Forward pass	Input → weights → activation → output
Loss function	Cross-entropy for classification, MSE for regression
Backpropagation	Gradient of loss w.r.t. every weight via chain rule
Adam optimizer	Adaptive learning rates per parameter — default choice

02. 1D CNNs for DNA

A 1D convolution slides a learnable filter across a sequence, detecting local patterns (motifs). Stacking multiple convolutional layers builds a hierarchy: layer 1 detects nucleotide patterns, layer 2 detects motif combinations, and so on.

Concept	Details
kernel_size	Length of the sliding filter window — typically 4–16 for DNA
out_channels	Number of distinct patterns the layer can detect
BatchNorm1d	Normalizes layer inputs — stabilizes training dramatically
AdaptiveMaxPool1d	Reduces any sequence length to a fixed output size

03. Transformers & Self-Attention

The attention mechanism lets every position in a sequence look at every other position. This global context is what makes Transformers so powerful for proteins — an amino acid at position 50 can directly influence position 300 in one forward pass.

Concept	Details
Query, Key, Value	Three projections of the same input — QK^T scores how much attention to pay
Multi-head attention	Run attention h times in parallel — each head learns different relationships
Positional encoding	Adds position information — Transformers have no inherent order
Layer norm	Normalize within each token — different from BatchNorm

04. ESM-2 Protein Language Model

Meta AI's ESM-2 is pre-trained on 250 million protein sequences. Like BERT for text, it learns deep representations of amino acid sequences that encode structural and functional information. Fine-tuning needs only hundreds of labeled examples.

Concept	Details
Masked language model	Predict masked amino acids — learns biological grammar
esm2_t6_8M	8M parameter version — runs on any laptop
esm2_t33_650M	650M version — near AlphaFold accuracy for many tasks
Mean pooling	Average embeddings over all positions → fixed-size sequence vector

■ CNN + Transformer DNA Classifier (PyTorch)

```
import torch, torch.nn as nn, torch.nn.functional as F
class DNAClassifier(nn.Module):
    """Hybrid CNN + Transformer for DNA classification"""
    def __init__(self, vocab=5, embed=64, num_classes=5):
        super().__init__()
        self.embed = nn.Embedding(vocab, embed)
        # CNN stem — local motif detection
        self.cnn = nn.Sequential(
            nn.Conv1d(embed, 128, 8, padding=4), nn.GELU(),
            nn.Conv1d(128, 256, 4, padding=2),   nn.GELU(),
            nn.AdaptiveMaxPool1d(64)
        )
        # Transformer — global context
        enc = nn.TransformerEncoderLayer(
            d_model=256, nhead=8, dim_feedforward=512,
            dropout=0.1, batch_first=True)
        self.transformer = nn.TransformerEncoder(enc, num_layers=2)
        # Classifier head
        self.head = nn.Sequential(
            nn.Linear(256, 128), nn.GELU(), nn.Dropout(0.3),
            nn.Linear(128, num_classes)
        )
    def forward(self, x):
        x = self.embed(x).permute(0,2,1)   # (B,64,L)
        x = self.cnn(x).permute(0,2,1)     # (B,64,256)
        x = self.transformer(x)            # (B,64,256)
        return self.head(x.mean(1))        # (B,5)
model = DNAClassifier()
print(f"Params: {sum(p.numel() for p in model.parameters()):,}")
```

■ Quiz Preview — Check Your Understanding

These are sample questions from the Moodle graded quiz for this week. Complete the full 10-question quiz in your course LMS after reviewing the material.

Q 1 **What does kernel_size=8 mean in a DNA CNN?**

✓ *The filter looks at 8 consecutive nucleotides at once*

Q 2 **Why use AdaptiveMaxPool?**

✓ *Outputs a fixed size regardless of input sequence length*

Q3 **In attention, what is Q·K^T / sqrt(d_k)?**

✓ *Scaled similarity score between every pair of positions*

Q4 **What is fine-tuning vs. training from scratch?**

✓ *Fine-tuning continues training a pre-trained model on your small dataset*

■ Resources & Next Steps

Type	Resource
■ Video	Course LMS → This week's video lesson (on-demand)
■ Dataset	Course LMS → datasets/ folder for this week
■ Project	Course LMS → project assignment notebook template
■ Quiz	Course LMS → Graded quiz (10 questions, timed)
■ Docs	See README.md in ai-biotech-portfolio GitHub repo