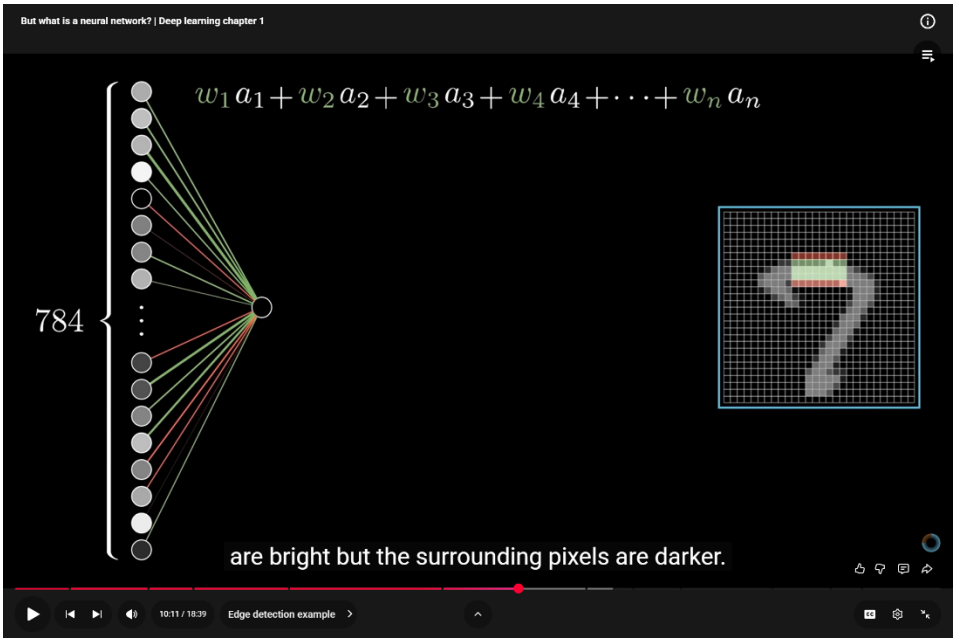
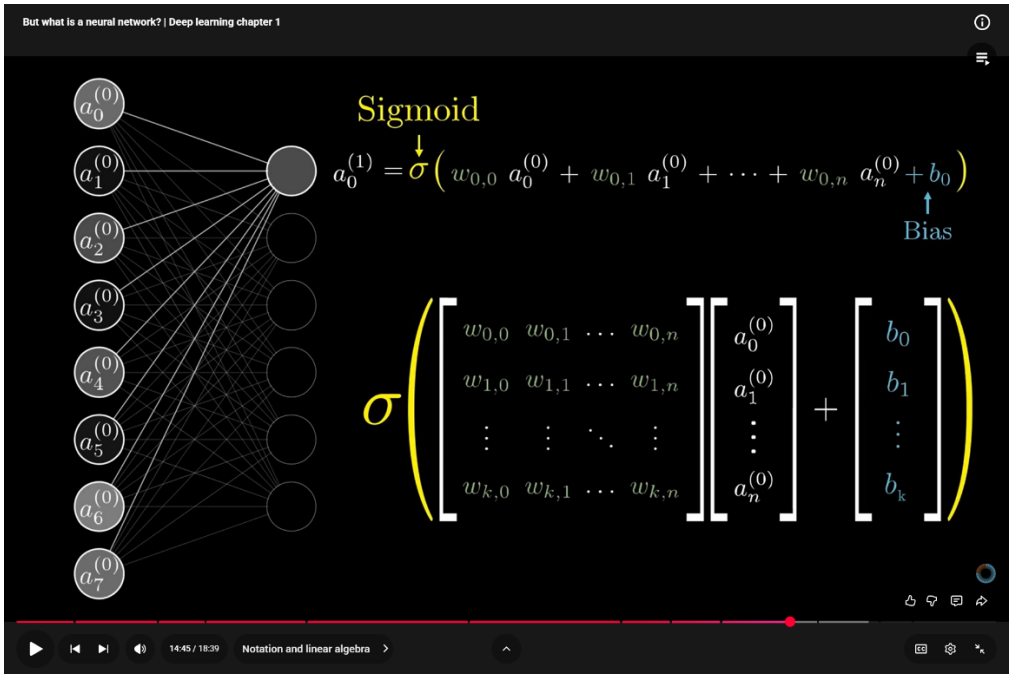
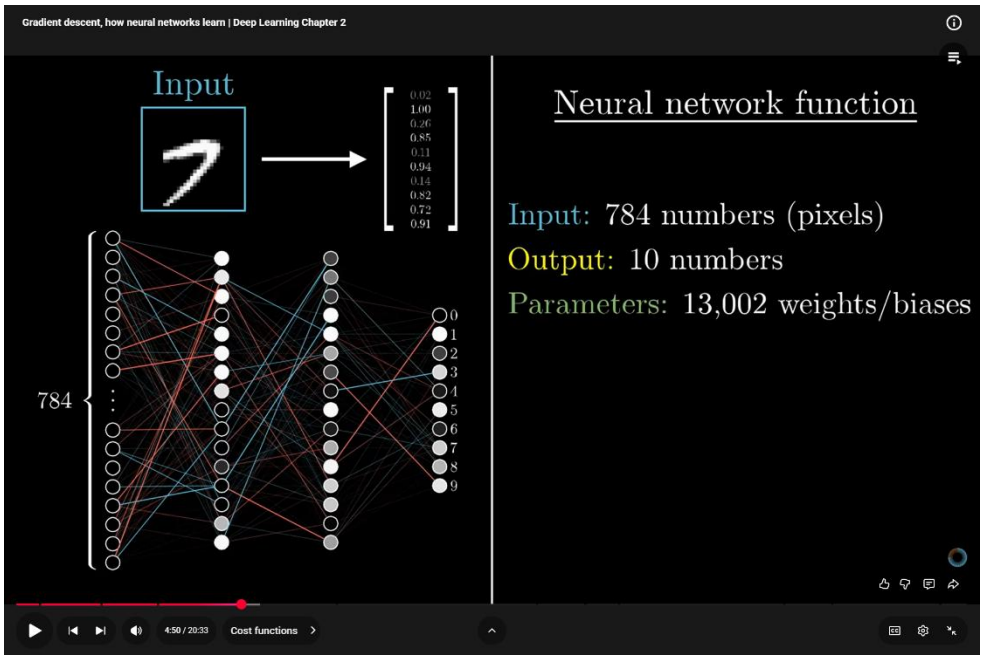


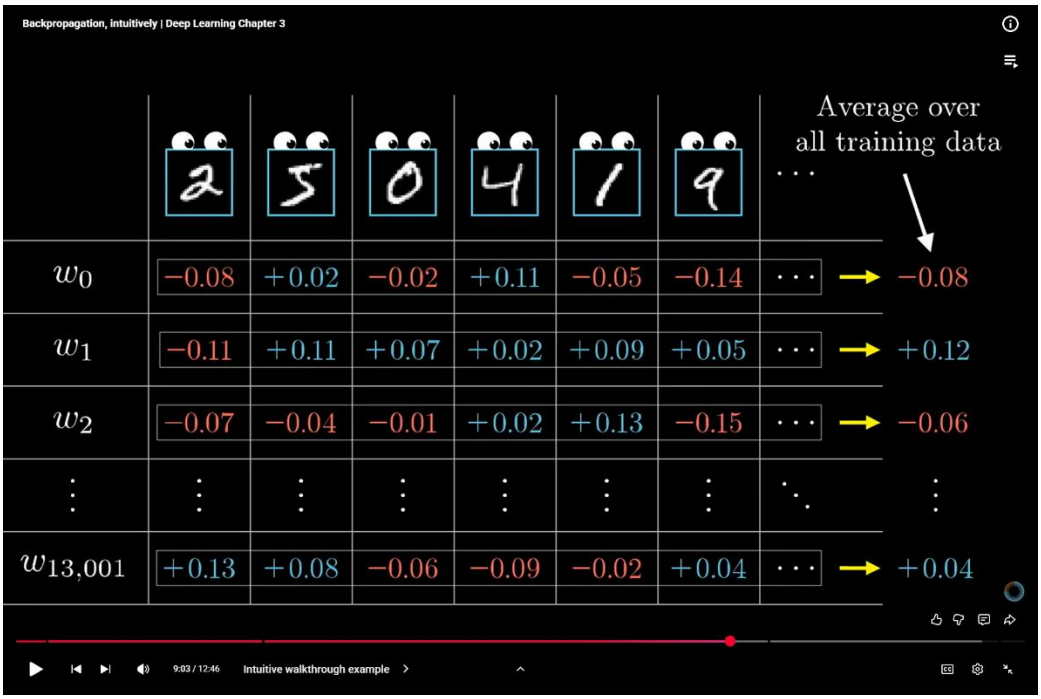
This screenshot helped me understand how a neural network processes image pixels. It clearly shows how the 784 input pixels of an MNIST image are connected to a single neuron through weighted sums $w_1 a_1 + w_2 a_2 + \cdots$. The visualization made it easier to see that each pixel contributes differently depending on its weight, which helped me understand how a network detects patterns like edges or strokes in handwritten digits.



This screenshot helped me understand how a neuron combines inputs using matrix multiplication and a bias term. It clearly shows how the weights, inputs, and biases fit together in one compact equation, and how the sigmoid activation is applied afterward. Seeing the visual connection between the neural network diagram and the corresponding linear algebra representation made the computation inside a layer much easier to understand.

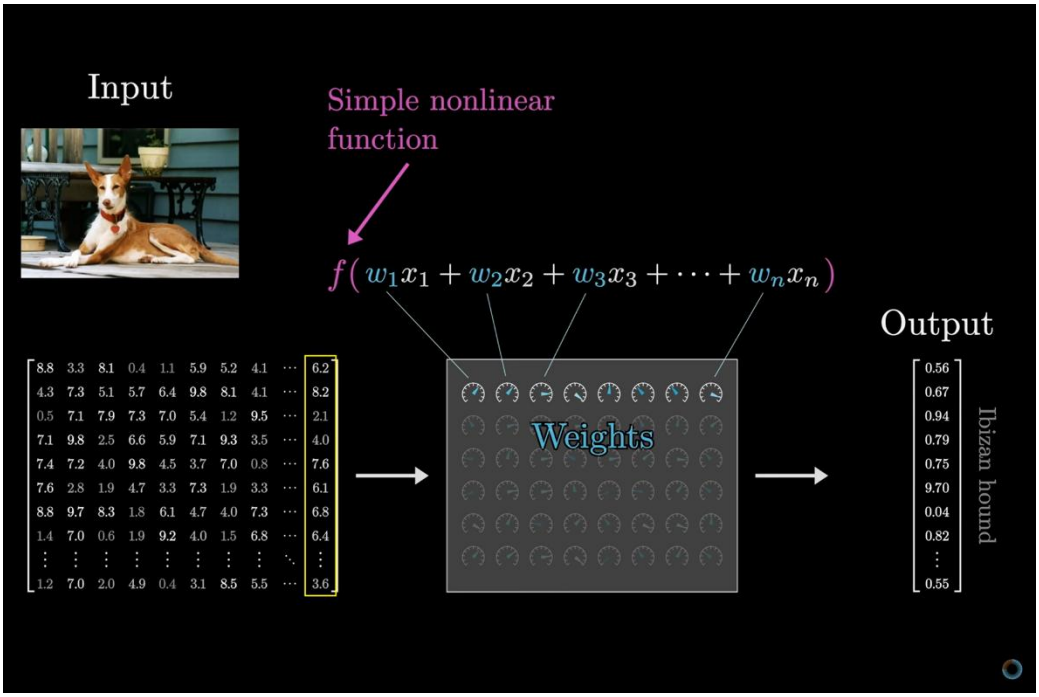


This screenshot helped me clearly understand the overall function of a neural network. It shows how a 28×28 image is flattened into 784 input values, processed through multiple hidden layers, and finally mapped to 10 output scores representing digit classes. The visualization of "13,002 parameters" also helped me appreciate how many weights and biases are involved even in a simple MNIST model.

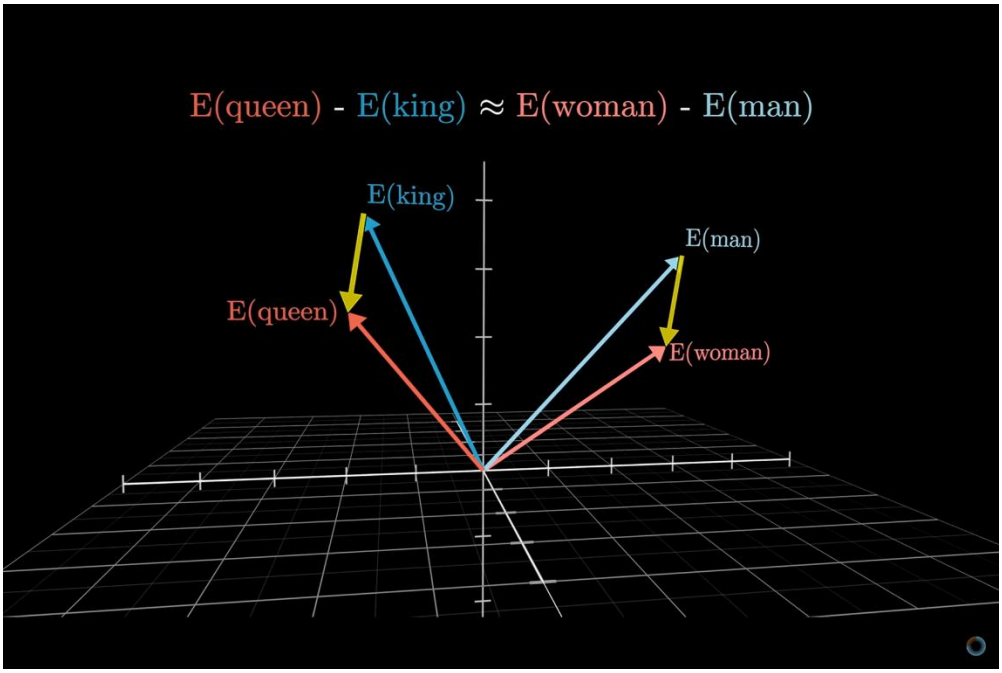


This screenshot helped me understand how backpropagation aggregates gradients across the entire training set. By showing the individual gradient contributions for each example and then averaging them, it visually clarified how each weight update is computed. The color-coded positive and negative values also made it easier to see how different samples push the weights in different directions.

This illustration clearly shows how previous models relied only on local context, while Transformers use attention to connect all words across a sentence. The comparison helped me understand why Transformers capture long-range meaning so effectively.



This illustration clearly shows how raw image pixels are converted into numerical features and processed through weighted nonlinear functions to generate class probabilities. It helped me understand the full pipeline from input image to output prediction.



This screenshot helped me understand how word embeddings capture semantic relationships through vector arithmetic. The parallel structure between the "king → queen" and "man → woman" directions made it clear that similar semantic shifts correspond to similar movements in embedding space. Seeing these vectors visually really clarified why analogies like *king* − *man* + *woman* ≈ *queen* work in practice.