***UNIT II:***

***[Text Book3]***

**Introducing Deep Learning: 1**

* **Biological and Machine Vision, 3**
  + Biological Vision 3
  + Machine Vision 8
  + The Neocognitron 8
  + LeNet-5 9
  + The Traditional Machine Learning Approach 12
  + ImageNet and the ILSVRC 13
  + AlexNet 14
  + TensorFlow Playground 17
* **Human and Machine Language**, 21
  + - Deep Learning for Natural Language Processing 21
    - Deep Learning Networks Learn Representations Automatically 22
    - Natural Language Processing 23
    - A Brief History of Deep Learning for NLP 24
  + Computational Representations of Language 25
    - One-Hot Representations of Words 25
    - Word Vectors 26
    - Word-Vector Arithmetic 29
    - word2viz 30
    - Localist Versus Distributed Representations 32
  + Elements of Natural Human Language 33
  + Google Duplex 35
* **Artificial Neural Networks,** 
  + The Input Layer 99
  + Dense Layers 99 A
  + Hot Dog-Detecting Dense Network 101
    - Forward Propagation Through the First Hidden Layer 102
    - Forward Propagation Through Subsequent Layers 103
  + The Softmax Layer of a Fast Food-Classifying Network 106
  + Revisiting Our Shallow Network 108
* **Training Deep Networks,** 
  + Cost Functions 111
    - Quadratic Cost 112
    - Saturated Neurons 112
    - Cross-Entropy Cost 113
  + Optimization: Learning to Minimize Cost 115
    - Gradient Descent 115
    - Learning Rate 117
    - Batch Size and Stochastic Gradient Descent 119
    - Escaping the Local Minimum 122
  + Backpropagation 124
  + Tuning Hidden-Layer Count and Neuron Count 125
  + An Intermediate Net in Keras 127
* **Improving Deep Networks.** 
  + Weight Initialization 131
    - Xavier Glorot Distributions 135
  + Unstable Gradients 137
    - Vanishing Gradients 137
    - Exploding Gradients 138
    - Batch Normalization 138
  + Model Generalization (Avoiding Overfitting) 140
    - L1 and L2 Regularization 141
    - Dropout 142
    - Data Augmentation 145
  + Fancy Optimizers 145
    - Momentum 145
    - Nesterov Momentum 146
    - AdaGrad 146
    - AdaDelta and RMSProp 146
    - Adam 147
  + A Deep Neural Network in Keras 147
  + Regression 149
  + TensorBoard 152

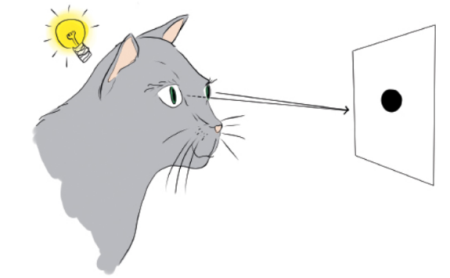
**Introducing Deep Learning: 1**

* **Biological and Machine Vision, 3**
  + Biological Vision 3

In modern mammals, a large proportion of the cerebral cortex—the outer, grey matter of the brain —is involved in visual perception. At Johns Hopkins University in the late 1950s, the physiologists David Hubel and Torsten Wiesel began carrying out their pioneering research on how visual information is processed in the mammalian cerebral cortex, work which contributed to them later being awarded a Nobel Prize. As depicted in Figure 1.4, Hubel and Wiesel conducted their research by showing images to anaesthetized cats while simultaneously recording the activity of individual neurons from the primary visual cortex, the first part of the cerebral cortex to receive visual input from the eyes.

Projecting slides onto a screen, Hubel and Wiesel began by presenting simple shapes like the dot shown in Figure 1.4 to the cats. Their initial results were disheartening: Their efforts were met with no response from the neurons of the primary visual cortex. They grappled with the frustration of how these cells, which anatomically appear to be the gateway for visual information to the rest of the cerebral cortex, would not respond to visual stimuli.

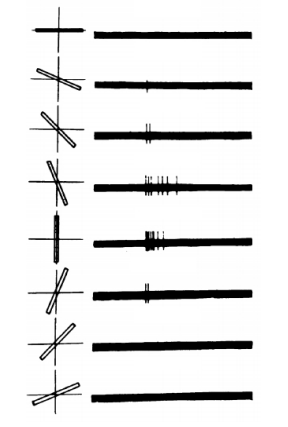
As they removed one of their slides from the projector, its straight edge elicited the distinctive crackle of their recording equipment to alert them that a primary visual cortex neuron was firing.



**Figure 1­4** Hubel and Wiesel used a light projector to present slides to anaesthesized cats while they recorded the activity of neurons in the cats’ primary visual cortex. In their experiments, electrical recording equipment was implanted within the cat’s skull.

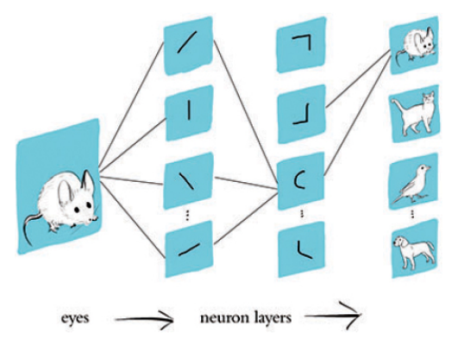
Through further experimentation, Hubel and Wiesel discovered that the neurons that receive visual input from the eye are in general most responsive to simple, straight edges. Fittingly then, they named these cells simple neurons.

As shown in Figure 1.5, Hubel and Wiesel determined that a given simple neuron responds optimally to an edge at a particular, specific orientation. A large group of simple neurons, with each specialized to detect a particular edge orientation, together are able to represent all 360 degrees of orientation. These edge ­orientation detecting simple cells then pass along information to a large number of so­ called complex neurons. A given complex neuron receives visual information that has already been processed by several simple cells so it is well­ positioned to recombine multiple line orientations into a more complex shape like a corner or a curve.



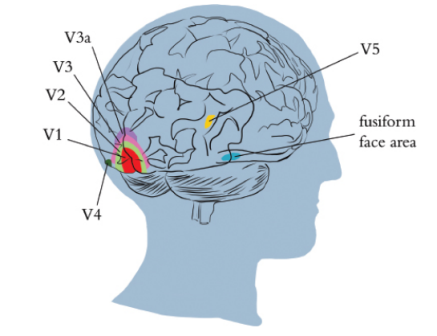
**Figure 1­5** A “simple” cell in the primary visual cortex of a cat fires at different rates, depending on the orientation of a line shown to the cat. The orientation of the line is provided in the left­hand column of the figure, while the right­hand column shows the firing (electrical activity) in the cell over time (one second). A vertical line (in the fifth row) causes the most electrical activity for this particular simple cell. Lines slightly off vertical (in the intermediate rows) cause less activity for the cell, while lines approaching horizontal (in the top­most and bottom­most rows) cause little to no activity.

Figure 1.6 illustrates how, via many hierarchically­organized layers of neurons feeding information into increasingly higher­order neurons, gradually more complex visual stimuli can be represented by the brain. The eyes are focused on an image of a rat’s head. Photons of light stimulate neurons located in the retina of each eye and this raw visual information is transmitted from the eyes to the primary visual cortex of the brain. The first layer of primary visual cortex neurons to receive this input—what Hubel and Wiesel termed simple cells—are specialized to detect edges (straight lines) at specific orientations. There would be many thousands of such neurons; for simplicity, we’re only showing four. In our caricature, we’re illustrating that neurons one, three, and four are activated by viewing the rat’s head. These three simple neurons relay that information to a subsequent layer, where complex cells assimilate the information about various edge orientations, enabling them to represent more complex visual stimuli, like the curvature of the rat’s head. As information is passed through several subsequent further layers, the complexity and abstractness of the visual stimuli that can be represented incrementally increases. As depicted by the far ­right layer of neurons, following many layers of such hierarchical processing, the brain is ultimately able to represent visual concepts as abstract as a rat, a cat, a bird or a dog.



**Figure 1­6** A caricature of how consecutive layers of biological neurons represent visual information in the brain of, e.g., a cat or a human.

Neuroscientists have pieced together a fairly high­resolution map of regions that are specialized to process particular visual stimuli, e.g., color, motion, faces (see Figure 1.7).

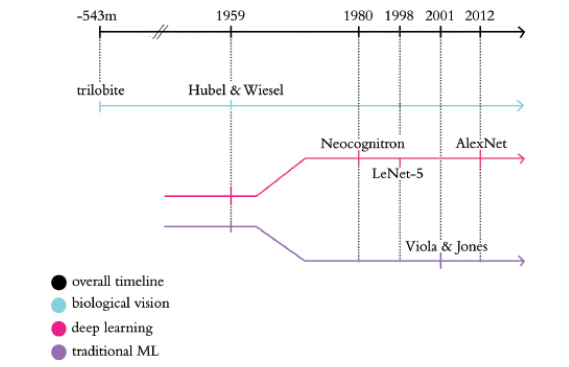


**Figure 1­7** Regions of the visual cortex. The V1 region receives input from the eyes and contains the “simple” cells that detect edge orientations. Through the recombination of information via myriad subsequent layers of neurons (including within the V2, V3, and V3a regions), increasingly abstract visual stimuli are represented. In the human brain (shown here), there are regions containing neurons with concentrations of specializations in, as examples, the detection of color (V4), motion (V5), and people’s faces (fusiform face area).

We covered the biological visual system primarily because it served as the inspiration for the modern deep learning approaches to machine vision

* + **Machine Vision 8**

Figure 1.8 provides a concise historical timeline of vision, in both biological organisms and machines. The top timeline, in blue, highlights the development of vision in trilobites. The machine vision timeline is split into two parallel streams to call attention to two alternative approaches. The middle timeline, in pink, represents the deep learning. The bottom timeline, in purple, meanwhile represents the traditional machine learning path to vision, which —through contrast —will clarify why deep learning is distinctively powerful and revolutionary.

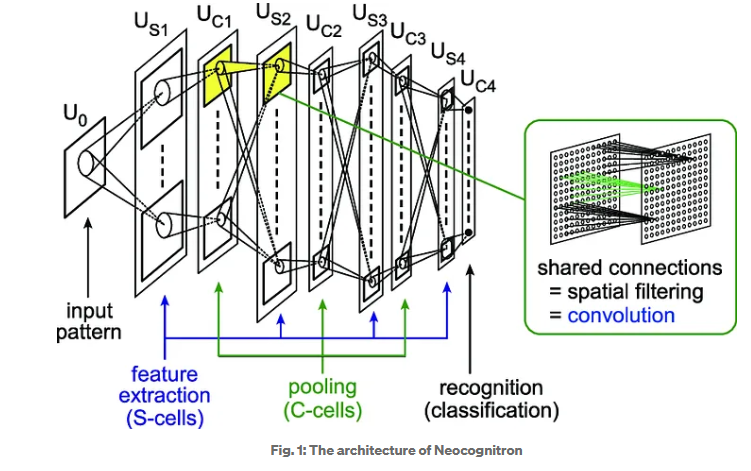


**Figure 1­8** Abridged timeline of biological and machine vision, highlighting the key historical moments in the deep learning and traditional machine learning approaches to vision

* + **The Neocognitron (1980) 8**

In the world of deep learning, Convolutional Neural Network (CNN) is a class of artificial neural network, most commonly used for image analysis. Since inception, CNN architectures have gone through rapid evolution and in recent years have achieved results which were previously considered possible only via human execution/intervention. Depending on the task at hand, and the corresponding constraints, a wide variety of architectures are available today. These are too deep to be completely visualized and are often treated as black boxes. But were they always like that? Isn’t it interesting to delve down into the history of CNN architectures? Tie your seatbelts for a quick trip into this history.

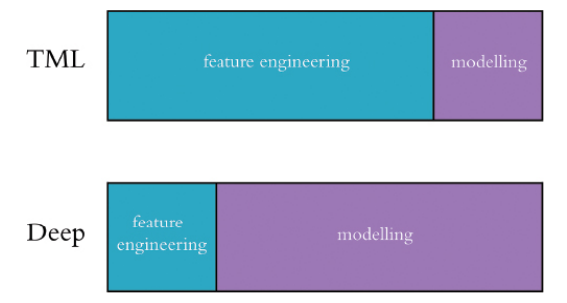
Neocognitron was the first architecture of its kind, perhaps the earliest precursor of CNNs. The concepts of feature extraction, pooling layers, and using convolution in a neural network were introduced and finally recognition or classification at the end was proposed in the Neocognitron. The structure of the network was inspired by that of the visual nervous system of vertebrates. In the whole network, with its alternate layers of S-cells (simple cells or lower order hypercomplex cells) and C-cells (complex cells or higher order hypercomplex cells), the process of feature-extraction by S-cells and toleration of positional shift by C-cells was repeated. During this process, local features extracted in lower stages are gradually integrated into more global features. It was used for handwritten (Japanese) character recognition and other pattern recognition tasks, and further paved the way for convolutional neural networks.

****

* + **LeNet-5 (1989–1998)9**

While the neocognitron was capable of, for example, identifying handwritten characters, the accuracy and efficiency of Yann LeCun and Yoshua Bengio’s LeNet­5 model made it a significant development. LeNet­5’s hierarchical architecture (Figure 1.12) built on Fukushima’s lead and the biological inspiration uncovered by Hubel and Wiesel. In addition, LeCun and his colleagues’ benefited from superior data for training their model, faster processing power and, critically, the backpropagation algorithm.

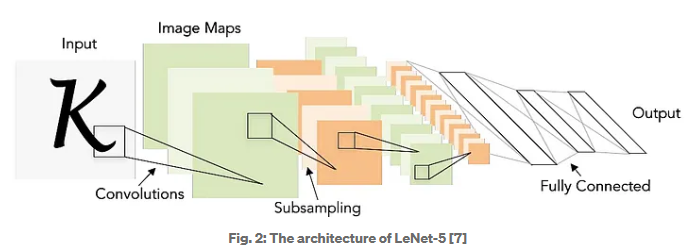
Backpropagation, often abbreviated to backprop, facilitates efficient learning throughout the layers of artificial neurons within a deep learning model. Together with their data and processing power, backprop rendered LeNet­5 sufficiently reliable to become an early commercial application of deep learning: It was used by the United States Postal Service to automate the reading of ZIP codes written on mail envelopes. In Chapter 10, on machine vision, we will experience LeNet­5 first­hand by designing it ourselves and training it to (guess what!) recognize handwritten digits. In LeNet­5, Yann LeCun and his colleagues had an algorithm that could correctly predict what handwritten digits had been drawn without them needing to include any expertise about handwritten digits in their code. As such, LeNet­5 provides an opportunity to introduce a fundamental difference between deep learning and the traditional machine learning ideology. As conveyed by Figure 1.13, the traditional machine learning (ML) approach is characterized by practitioners investing the bulk of their efforts into engineering features. This feature engineering is the application of clever, and often elaborate, algorithms to raw data in order to preprocess them into input variables that can be readily modeled by traditional statistical techniques. These techniques—e.g., regression, random forest, support vector machine—are seldom effective on unprocessed data, and so the engineering of input data has historically been a prime focus of machine learning professionals.



**Figure 1­13** Feature engineering—the transformation of raw data into thoughtfully­transformed input variables—often predominates the application of traditional machine learning algorithms. In contrast, the application of deep learning often involves little to no feature engineering, with the majority of time spent instead on the design and tuning of model architectures.

In general, a minority of the traditional ML practitioner’s time is spent optimizing ML models or selecting the most effective one from those available. The deep learning approach to modeling data turns these priorities upside­down. The deep learning practitioner typically spends little to none of her time engineering features, instead spending it modeling data with various artificial neural network architectures that process the raw inputs into useful features automatically. T

The name convolutional neural networks actually originated with the design of the LeNet by Yann LeCun and team ([paper](http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf)). It was largely developed between 1989 and 1998 for the handwritten digit recognition task.

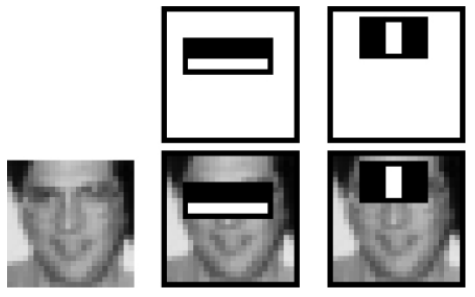


The overall architecture was [CONV-POOL-CONV-POOL-**FC**-**FC**]. It used 5x5 convolution filters with a strike of 1. The pooling (subsampling) layers were 2x2 with a stride of 2. It has about 60 K parameters.

* + **The Traditional Machine Learning Approach 12**

To make clear what feature engineering is, Figure 1.14 provides a celebrated example from Paul Viola and Michael Jones in the early noughties. Viola and Jones employed rectangular filters such as the vertical or horizontal black and white bars shown in the figure. Features generated by passing these filters over an image can be fed into machine learning algorithms to reliably detect the presence of a face. Their work is notable because the algorithm was efficient enough to be the first real­ time face detector outside the realm of biology.

Devising clever face­ detecting filters to process raw pixels into features for input into a machine learning model was accomplished via years of research and collaboration on the characteristics of faces. And, of course, it is limited to detecting faces in general, as opposed to being able to recognize a particular face as, say, Angela Merkel’s or Oprah Winfrey’s. To develop features for detecting Oprah in particular, or for detecting some non ­face class of objects like houses, cars, or Yorkshire Terriers, would require the development of expertise in that category, which could again take years of academic ­community collaboration to execute both efficiently and accurately. If only we could circumnavigate all that time and effort somehow…

****

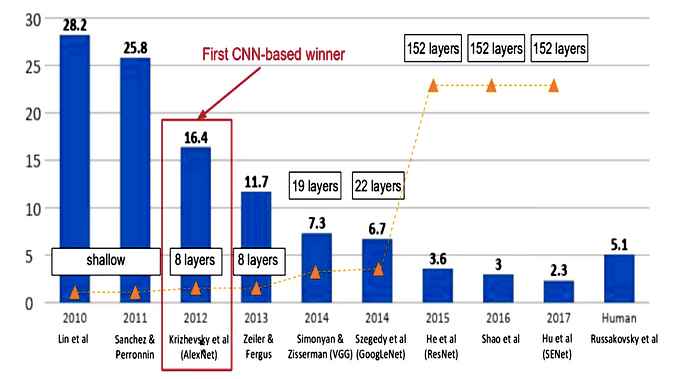
**Figure 1­14** Engineered features leveraged by Viola and Jones (2001) to detect faces reliably. Their efficient algorithm found its way into FujiFilm cameras, facilitating real­ time auto­focus.

* + **ImageNet and the ILSVRC 13**

As mentioned earlier, one of the advantages LeNet­5 had over the neocognitron was a larger, high­ quality set of training data. The next breakthrough in neural networks was also facilitated by a high­ quality public dataset—this time much larger: ImageNet, a labelled index of photographs devised by Fei­Fei Li (Figure 1.15), armed machine vision researchers with an immense catalog of training data. For reference, the handwritten digit data used to train LeNet­5 contained tens of thousands of images. ImageNet, in contrast, contains tens of millions.

The fourteen million images in the ImageNet data set are spread across 22,000 categories. These categories are as diverse as container ships, leopards, starfish and elderberries. Since 2010, Professor Li has run an open challenge called ILSVRC on a subset of the ImageNet data that has become the premier ground for assessing the world’s state ­of­ the ­art machine vision algorithms. The ILSVRC subset consists of 1.4 million images across a thousand categories. In addition to providing a broad range of categories, many of the selected categories are breeds of dogs, thereby evaluating the algorithms’ ability not only to distinguish broadly­ varying images, but also to specialize in distinguishing subtly varying ones.

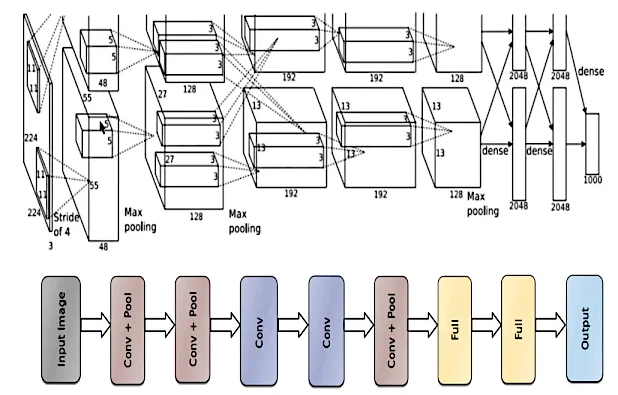
The credit for newer architectures of CNNs goes to [ImageNet](https://image-net.org/index.php" \t "_blank) (a dataset) classification challenge named ‘ImageNet large scale visual recognition challenge (ILSVRC)’. It was started in 2010 which led to a significant effort across researchers to benchmark their machine learning and computer vision models, in particular for image classification, on a common dataset. Performance was measured in Top-1 error and Top-5 error. In 2010, the winning error rate was 28.2% and it was done without neural networks. In 2011 researchers improved the score from 28.2% to 25.8% error rate. Fig. 3 shows all the winners with the corresponding errors as bar.



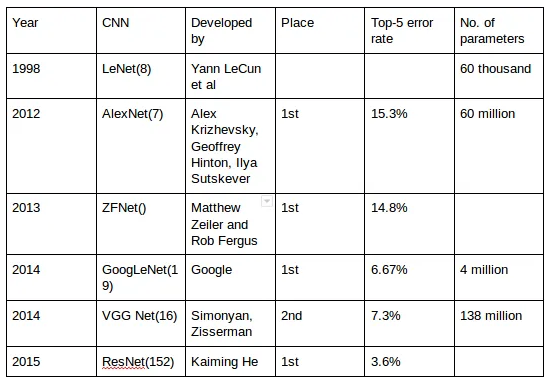
Finally in 2012, Alex Krizhevsky and Geoffrey Hinton came up with a CNN architecture popular to this day as AlexNet, which reduced the error from 25.8% to 16.4% which was a significant improvement at that time.

* + **AlexNet (2012)14**

AlexNet was the first winner of the ImageNet challenge and was based on a CNN, and since 2012, every year’s challenge has been won by a CNN; significantly outperforming other deep and shallow ( traditional) machine learning methods.

****

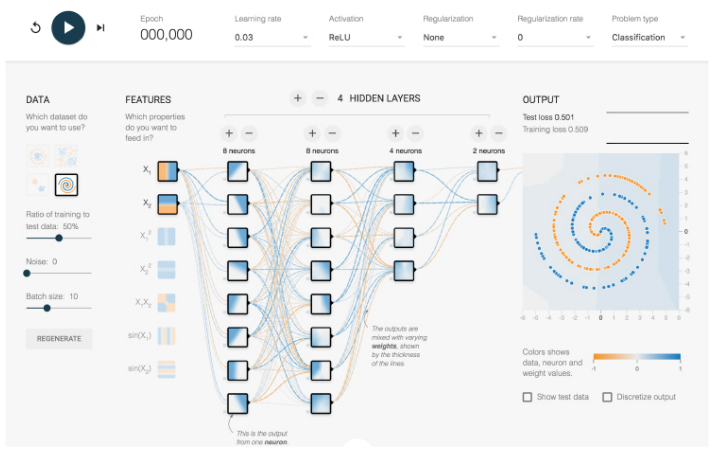
AlexNet has 8 layers in total (5 convolutional layers plus 3 fully connected layers), obviously trained on ImageNet Dataset. A normalization layer called the response normalization layer was first introduced. It normalized all the values in a particular location across the channels in a given layer. Further, it also introduced the rectified linear unit (ReLU) as an activation function. It has about 60 M parameters (Can you recall the number of parameters in LeNet?). Interestingly the convolutional layers cumulatively contain about 90–95% of computation but only about 5% of the parameters.



* + **TensorFlow Playground 17**

For a fun, interactive way to crystallize the hierarchical, feature­ learning nature of deep learning, make your way to the TensorFlow Playground via the following URL: bit.ly/TFplayground.

By using this custom link, your network should automatically look similar to the one shown in Figure 1.19. We’ll be returning to define all of the terms on the screen in Part II; for the present exercise, they can be safely ignored. It suffices at this time to know that this is a deep learning model. The model architecture consists of six layers of artificial neurons: an input layer on the left (below the FEATURES heading), four “hidden” layers (which bear the responsibility of learning), and an output layer (the grid on the far right ranging from 6 to +6 on both axes). The network’s goal is to learn how to distinguish orange dots (negative cases) from blue dots (positive cases) based solely on their location on the grid. As such, in the input layer, we are only feeding in two pieces of information about each dot: its horizontal position (X ) and its vertical position (X ). The dots that will be used as training data are shown by default on the grid. By clicking the Show test data toggle, you can also see the location of dots that will be used to assess the performance of the network as it learns. Critically, these test data are not available to the network while it’s learning, so they help us ensure that the network generalizes well to new, unseen data.

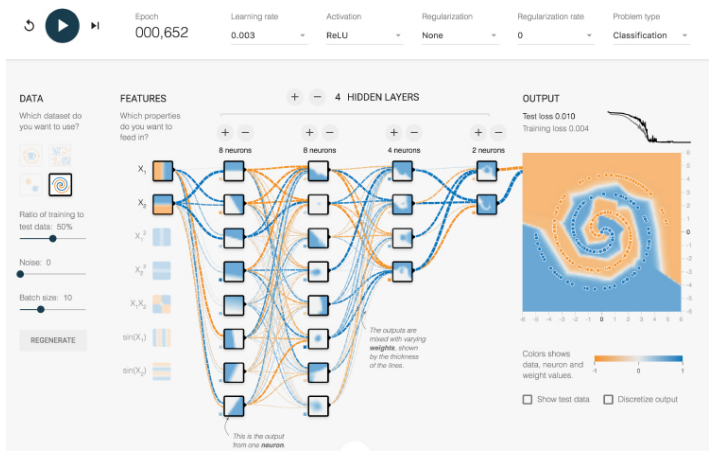


**Figure 1­19** A deep neural network ready to learn how to distinguish a spiral of orange dots (negative cases) from blue dots (positive cases) based on their position on the X and X axes of the grid on the right. Click the prominent Play arrow in the top­ left corner. Enable the network to train until the “Training loss” and “Test loss” in the top ­right corner have both approached zero, say less than 0.5. How long this takes will depend on the hardware you’re using but will hopefully not be more than a few minutes.

As captured in Figure 1.20, you should now see the network’s artificial neurons representing the input data with increasing complexity and abstraction the deeper (further to the right) they are positioned—as in the neocognitron, LeNet­5, and AlexNet. Every time the network is run, the neuron­ level details of how the network solves the spiral classification problem are unique, but the general approach remains the same.

The artificial neurons in the left­most “hidden” layer are specialized in distinguishing edges (straight lines), each at a different particular orientation. Neurons from the first hidden layer pass information to neurons in the second hidden layer, each of which recombine the edges into slightly more complex features like curves. The neurons in each successive layer recombine information from the neurons of the previous layer, gradually increasing the complexity and abstraction of the features they can represent.

By the final (right­most) layer, the neurons are adept at representing the intricacies ofthe spiral shape, enabling the network to accurately predict whether a dot is orange (a negative case) or blue (a positive case) based on its position (X and X coordinates) in the grid. Hover over a neuron to project it onto the far ­right OUTPUT grid and examine its individual specialization in detail.



**Figure 1­20** The network after training