✍️

Handwritten Digit Recognition with a Back-Propagation Network

Author	Yann LeCun
Link	https://proceedings.neurips.cc/paper/1989/file/53c3bce66e43be4f209556518c2fcb54-Paper.pdf
Completed	@June 27, 2023
Publisher	Neurips 1989
Status	Done

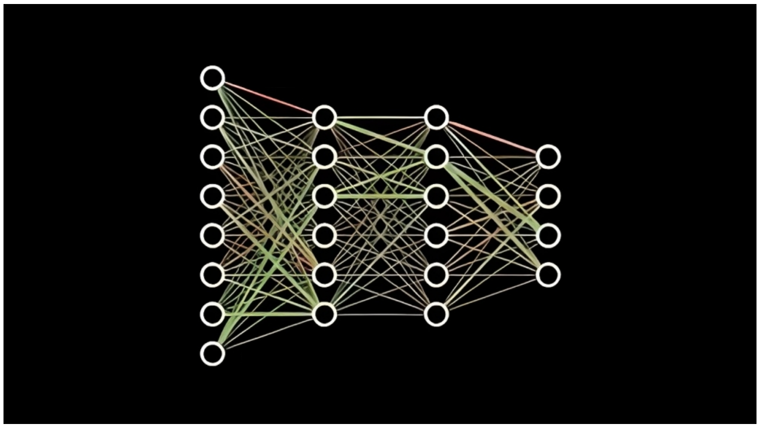

Table of contents

Preprocessing

Paragraph 1:

- This passage is discussing the process of analyzing postal codes using computer algorithms. Postal Service contractors, specifically Wang and Srihari in 1988, were responsible for performing several steps in this process.

- The first step is acquisition, which involves capturing the postal code image data. Binarization is the process of converting the image into a binary format, where each pixel is represented as either black or white. The purpose of binarization is to enhance the readability of the postal code.

- Once the image is binarized, the next step is to locate the zip code within the image. This involves identifying the specific area where the zip code is located.
- After locating the zip code, preliminary segmentation takes place. Segmentation refers to the process of separating individual digits from one another. Ideally, if each character is separate and not connected to its neighbors, segmentation would be a relatively simple task. However, in practice, characters can be connected or have ambiguous shapes, making segmentation more challenging. Mis-segmentation can lead to errors in recognizing characters, as shown in Figure 2, where broken 5's are observed as a common example.

Paragraph 2:

- In this context, the passage is explaining how the size of a digit (such as a number) in an image is standardized to make it suitable for input into a back-propagation network, which is a type of machine learning algorithm.
- The size of the digit image can vary, but it is typically around 40 by 60 pixels. However, the back-propagation network requires a fixed-size input. To address this, a process called normalization is performed to resize the characters.
- Normalization involves applying a linear transformation to the original image to make the characters fit within a 16 by 16 pixel space. This transformation maintains the original aspect ratio of the character, meaning it keeps the same proportions and shape. Before the transformation, any unnecessary marks or elements in the image are removed.
- As a result of the linear transformation, the resulting image is no longer strictly black and white (binary). Instead, it contains multiple gray levels, indicating different shades of gray. This is because a variable number of pixels from the original image are now mapped to a single pixel in the target image.
- To ensure consistency, the gray levels in each image are further adjusted. They are scaled and translated so that the values fall within the range of -1 to 1. This scaling and translation process helps standardize the range of pixel values across different digit images, making it easier for the back-propagation network to learn and make accurate predictions.

The Network

Paragraph 1:

- The connections within the network are adaptive, meaning they can change and adjust themselves based on the data they receive. However, these connections have certain limitations or constraints imposed on them. The training of the network is performed using a method called back-propagation, which helps the network learn and improve its accuracy over time.

- This approach differs from earlier work (referred to as Denker et al., 1989) where the initial connections in the network were manually set to specific values. In this case, all the connections in the network are adjusted through training.
- The input to the network is a normalized image that has been resized to a fixed size of 16 by 16 pixels. The output of the network consists of 10 units, with each unit representing a different class or category. For example, if we are trying to recognize a digit from 0 to 9, there would be 10 output units.
- When an image of a particular digit is presented to the network, the desired output is +1 for the corresponding output unit representing that class and -1 for the other output units representing different classes. This way, the network learns to associate specific patterns with the correct class.

Paragraph 2:

- In machine learning, we use neural networks to recognize and classify different patterns. One challenge is that a fully connected network, where each neuron is connected to every other neuron, can have too many parameters to work effectively.
- To address this, we use a restricted connection-scheme, meaning we limit the connections in the network based on our prior knowledge about shape recognition. We know that shape recognition benefits from detecting and combining local features, which are specific patterns or characteristics within an image.
- To achieve this, we introduce the concept of a feature map. Imagine a grid of small regions or "neurons" that cover the input image. Each neuron in the feature map is responsible for detecting a specific local feature. For example, one neuron might focus on detecting curves, while another neuron focuses on detecting corners.
- Instead of connecting every neuron in the feature map to the entire input image, we use a local receptive field. This means that each neuron only considers a small region of the image. By doing this, the network focuses on local information rather than trying to analyze the entire image at once.
- The feature map neurons perform a computation on their receptive fields, typically using a small-sized kernel (a sort of filter) and a squashing function. This computation helps highlight the presence of the specific local feature they are responsible for.
- To reduce the number of parameters and improve computational efficiency, we use weight sharing. This means that multiple neurons in a feature map share the same weights, meaning they perform the same computation but on different parts of the image. This helps greatly reduce the complexity of the network.
- Weight sharing has another benefit: it introduces shift invariance. Shift invariance means that if we shift the input image (for example, move it slightly to the left or right), the result on the feature

map will shift accordingly, but the overall recognition won't be affected. This is because the local features we are detecting are not dependent on their absolute position in the image.

- In practice, we often use multiple feature maps, each focusing on different local features. These feature maps work together to extract and identify various patterns and shapes within the image. This approach enhances the network's ability to recognize shapes accurately and robustly.
- Overall, by using a restricted connection-scheme with local feature detection and weight sharing, we can build a neural network that efficiently recognizes shapes while reducing complexity and improving shift invariance.

Paragraph 3:

- In a neural network, we can apply the concept of local, convolutional feature maps to subsequent hidden layers. This means that we can use the same idea of detecting local features as we discussed before, but in deeper layers of the network.
- As we move to higher-level features, the extracted features become more complex and abstract. Interestingly, these higher-level features are less dependent on the precise location of the features in the input image. This is actually beneficial because it means that slight distortions or translations of the input image will have a reduced effect on the representation of these features.
- To introduce this invariance to distortions and translations, each feature extraction in our network is followed by an additional layer that performs local averaging and subsampling. In simpler terms, this layer takes the feature map and reduces its resolution by summarizing and condensing the information within small regions.
- By reducing the resolution, we achieve a certain level of invariance to distortions and translations. This means that even if the input image is slightly distorted or shifted, the higher-level features can still be recognized effectively.
- In our network, a functional module consists of a layer of shared-weight feature maps followed by an averaging/subsampling layer. This combination is similar to the Neocognitron architecture, a neural network model introduced in the past. However, there is a notable difference: we use backpropagation (a learning algorithm) instead of unsupervised learning, which we believe is more appropriate for solving classification problems like the one we are working on.
- To summarize, as we go deeper into the network, we extract more complex features using the same local feature detection approach. Higher-level features are less dependent on precise location and can tolerate slight distortions. To enhance this tolerance, we introduce an additional layer that reduces the resolution and provides invariance to distortions and translations. This approach, along with backpropagation, helps us build a network that is effective in recognizing and classifying patterns.

Paragraph 4:

- The network consists of four hidden layers: HI, H2, H3, and H4. Each of these layers serves a specific purpose in processing the input data.
- Layers HI and H3 are called shared-weights feature extractors. This means that these layers are responsible for detecting and extracting local features from the input data. They use a technique called weight sharing, which allows multiple units in the same layer to share the same set of weights. This weight sharing helps reduce the number of parameters in the network and improves efficiency.
- On the other hand, layers H2 and H4 are averaging/subsampling layers. These layers come after the shared-weights feature extractors and perform two operations: averaging and subsampling. Averaging involves taking the average of a small region of the feature map, which helps to condense the information and reduce the resolution. Subsampling means reducing the size of the feature map by selecting a subset of values. These operations further enhance the network's ability to handle distortions and translations by introducing a level of invariance.

Paragraph 5:

- In the given description, we are focusing on the layer called H1 (hidden layer 1) in the network architecture. Here's what it means:

1. Input Size: The active part of the input, which contains the information we are interested in, is a 16 by 16 square. However, to avoid any issues when a kernel (a small filter) overlaps the boundary of this square, the actual input size is slightly larger at 28 by 28.
2. Organization of H1: The H1 layer is composed of 4 groups, and each group contains 576 units. These 4 groups are arranged as 4 independent 24 by 24 feature maps. So, we can think of H1 as having four sets of feature maps.
3. Feature Maps: Each feature map in H1 is designated by a label, such as H1.1, H1.2, H1.3, and H1.4. Each unit within a feature map takes its input from a 5 by 5 neighborhood on the input plane. In other words, it considers a small square region of size 5 by 5 from the input.
4. Weight Sharing: The connections between units in a given feature map have a special property called weight sharing. It means that all units within a feature map share the same set of weights, including a bias term. For example, all 576 units in feature map H1.1 use the exact same set of 26 weights, and units in other feature maps (H1.2, H1.3, and H1.4) have their own separate set of weights.

- To summarize, in the H1 layer of the network, we have 4 groups of feature maps, each containing 576 units. The feature maps are organized as 4 independent 24 by 24 maps. Each unit within a feature map takes input from a 5 by 5 neighborhood on the input plane. Weight sharing is employed, which means all units within a feature map use the same set of weights. This arrangement allows the network to capture different features and patterns from the input data effectively.

Paragraph 5:

- Composition of Layer H2: Layer H2 is referred to as the averaging/subsampling layer. It consists of 4 planes, each having a size of 12 by 12. These planes can be thought of as separate grids.
- Input Relationship with Layer H1: Each unit in one of the planes of H2 takes inputs from 4 units on the corresponding plane in H1. In other words, there is a one-to-one relationship between the units in H2 and a group of 4 units in H1.
- Non-Overlapping Receptive Fields: The receptive fields in H2 are designed not to overlap. A receptive field is the specific region in the input that a unit in the subsequent layer receives information from. So, in H2, each unit's receptive field is distinct and does not overlap with the others.
- Equal Weight Constraints: All the weights within a single unit in H2 are constrained to be equal. This means that the connections between the units in H2 and the corresponding units in H1 have the same weight values. This constraint ensures that each unit in H2 performs a consistent operation on its inputs.
- Local Averaging and Subsampling: With the equal weights and non-overlapping receptive fields, layer H2 performs a local averaging and subsampling of the information from H1. This means that each unit in H2 computes an average of its inputs and reduces the spatial resolution of the feature maps from H1 by a factor of 2 in each direction (2 to 1 subsampling).
- In summary, layer H2 consists of 4 planes, each sized 12 by 12. It takes inputs from corresponding units in layer H1. The receptive fields in H2 do not overlap, and the weights within a unit are equal. The layer performs a local averaging operation on the inputs and reduces the spatial resolution by subsampling. This arrangement helps extract and condense important information from the previous layer, H1.

Paragraph 6:

- Let's break down the description of layer H3:

1. Composition of Layer H3: Layer H3 is made up of 12 feature maps, designated as H3.1, H3.2, H3.3, and so on. Each feature map consists of a 8 by 8 plane, with a total of 64 units in each map. So, in total, layer H3 has 12 feature maps.
2. Connection Scheme with Layer H2: The connection scheme between H2 and H3 is similar to the one between the input (HI) and H2, but slightly more complicated because H3 has multiple 2-D maps. Each unit in H3 receives inputs from one or two 5 by 5 neighborhoods centered around units that have the same positions within each H2 map.

3. Identical Weight Vectors within a Feature Map: Just like in previous layers, all the units within a feature map in H3 are constrained to have the same weight vectors. This means that the connections between H2 and H3 are organized in a way that ensures each unit within a feature map in H3 shares the same weights with the corresponding units in H2.
4. Input Selection Scheme: The feature maps in H3 determine which feature maps in H2 they receive inputs from based on a specific scheme described in table 1. This scheme dictates the connectivity pattern between H2 and H3, ensuring that each map in H3 selectively chooses its inputs from specific maps in H2.

- Moving on to layer H4:

1. Composition of Layer H4: Layer H4 is composed of 12 groups, each containing 16 units arranged in a 4 by 4 plane. So, there are a total of 12 groups in H4, each representing a separate set of units.
2. Role Similar to Layer H2: Layer H4 plays a similar role to layer H2. It performs local averaging and subsampling operations on the inputs it receives, just like H2 does. The composition of H4 allows it to condense and abstract information from the previous layer, H3.

- In summary, layer H3 consists of 12 feature maps, each containing 64 units arranged in an 8 by 8 plane. The connections between H2 and H3 are organized based on specific neighborhoods and weight sharing. Layer H4 is composed of 12 groups, each with 16 units arranged in a 4 by 4 plane, and it performs operations similar to H2. Together, these layers help extract and process features of increasing complexity and abstraction from the input data.

Paragraph 7:

- In the described network architecture, the output layer is the final layer of the network. It consists of 10 units, and each unit is connected to every unit in layer H4. This means that there is a direct connection between each unit in the output layer and each unit in layer H4.
- To summarize the network's composition, it consists of a total of 4,635 units. These units are distributed across the input layer, hidden layers (H1, H2, H3, H4), and the output layer. The connections between these units create a total of 98,442 connections in the network.
- When considering the number of independent parameters, the network has a total of 2,578. Parameters refer to the weights and biases associated with the connections between the units. These parameters determine the network's behavior and are adjusted during the training process to optimize its performance.
- The described architecture was derived using a technique called Optimal Brain Damage, which is a method for optimizing neural network architectures by pruning unnecessary parameters. The initial architecture, which had four times more parameters, served as a starting point. Through the application of the Optimal Brain Damage technique, the architecture was refined to reduce the number of parameters while maintaining or improving its performance.

Results

Paragraph 1:

- After training the neural network for 30 passes (or iterations) using a training set consisting of 7,291 handwritten digits and 2,549 printed digits, the network achieved an error rate of 1.1%. The error rate represents the percentage of incorrectly classified digits in the training set. Additionally, the Mean Squared Error (MSE), which measures the average squared difference between the predicted and actual outputs, was calculated to be 0.017. A lower MSE indicates better accuracy of the network's predictions.
- The network's performance was also evaluated on a separate test set, which comprised 2,007 handwritten characters and 700 printed characters. In this test set, the network achieved an error rate of 3.4%, meaning that 3.4% of the characters were classified incorrectly. The MSE on the test set was calculated to be 0.024.
- It is important to note that all the classification errors occurred specifically on the handwritten characters, indicating that the network had more difficulty accurately identifying and classifying the handwritten digits compared to the printed ones.

Paragraph 2:

- In a practical application, the focus is not only on the raw error rate but also on the number of rejections required to achieve a desired level of accuracy. In this study, the researchers measured the percentage of test patterns that needed to be rejected in order to achieve a 1% error rate. Rejection criteria were based on three conditions: the activity level of the most-active output unit, the activity level of the second most-active unit, and the difference between their activity levels, all compared against specific thresholds (t1, t2, and td).
- The best result obtained on the complete test set was a 5.7% rejection rate to achieve a 1% error rate. This means that approximately 5.7% of the test patterns needed to be rejected by the network because they did not meet the specified criteria. When considering only the handwritten set, the rejection rate was higher at 9% for the same 1% error rate.
- It's worth noting that the thresholds for rejection were determined based on performance measures on the test set. It is important to mention that about half of the substitution errors in the testing set were caused by faulty segmentation, where the network had difficulty separating individual characters. Additionally, a quarter of the errors were due to incorrect assignment of the desired category. Some of the remaining misclassifications were attributed to ambiguous images that even humans found challenging to interpret, and in a few cases, the network made errors without any clear reason.
- Overall, the percentage of rejections provides insights into the network's ability to filter out uncertain or ambiguous inputs in order to achieve a desired level of accuracy.

Paragraph 3:

- In this study, a second-order version of back-propagation was employed for the learning process. It is notable that despite using a second-order approach, the learning phase only required 30 passes through the training set. This efficient learning can be attributed to the substantial redundancy present in real data.
- The term "redundancy" here refers to the fact that real-world data often contains patterns and similarities that can be leveraged for learning. This redundancy allows the network to generalize and make accurate predictions even with a relatively small number of training iterations.
- Additionally, it's mentioned that a complete training session, which includes the 30 passes through the training set as well as testing, took approximately three days to complete. This was performed using a SUN SPARCstation 1 computer equipped with the SN2 connectionist simulator. The time required for training reflects the computational complexity of the learning algorithm and the size of the neural network being trained.
- Overall, the efficiency of the learning process in terms of both the number of passes required and the computational time highlights the effectiveness of the network architecture and the ability of the second-order back-propagation algorithm to effectively learn from the available data.

Paragraph 4:

- After the training phase, the trained neural network was implemented on a commercial Digital Signal Processor (DSP) board that contained an AT&T DSP-32C general-purpose DSP chip. This chip is capable of performing 12.5 million multiply-add operations per second on 32-bit floating-point numbers. The DSP chip operates as a coprocessor within a PC system that is connected to a video camera.
- In this implementation, the PC is responsible for tasks such as digitization (converting the image into a digital format), binarization (converting the image into a binary representation), and segmentation (identifying and separating individual digits within the image). These preprocessing steps are important for preparing the image data for input into the neural network.
- Once the preprocessing is completed, the DSP chip takes over and performs the size-normalization of the digits. This step ensures that all the digits have a consistent size by transforming them into a standardized format. After normalization, the DSP chip carries out the classification of the digits using the trained neural network.
- The overall throughput, or the rate at which the system can process classifications, is reported to be around 10 to 12 classifications per second. This includes the time required for image acquisition, digitization, preprocessing, size-normalization, and classification. The main limiting factor in the overall throughput is the normalization step, which is computationally intensive.

However, once the digits are normalized, the DSP chip can perform more than 30 classifications per second, indicating its high processing speed for the classification task.

- This implementation demonstrates the division of tasks between a PC and a DSP chip to efficiently process and classify handwritten digits in real-time applications. The PC handles the initial image processing steps, while the DSP chip with its high-speed processing capabilities performs the size-normalization and classification tasks, enabling rapid and accurate recognition of handwritten digits.

Conclusion

- The successful application of back-propagation learning in this study has achieved state-of-the-art results in recognizing handwritten digits, which is a challenging real-world task. The neural network used in this study had a large number of connections but relatively few free parameters, making it an efficient and effective architecture for the task.
- One key advantage of this network architecture is that it incorporates geometric knowledge about the task into the system. This means that the network can learn and recognize handwritten digits directly from a low-level representation of the data, without the need for complex and elaborate feature extraction techniques.
- Despite the large size of the training set, the learning time was relatively short due to the redundant nature of the data and the constraints imposed on the network. This indicates that the network was able to efficiently learn from the data and generalize well to new examples.
- The scalability of the network was also impressive, surpassing expectations based on smaller artificial problems typically used to evaluate back-propagation. This suggests that the network's performance is not limited to the specific task of handwritten digit recognition and could potentially be extended to larger tasks, such as recognizing alphanumeric characters.
- Furthermore, the final network architecture and learned weights obtained through back-propagation were easily implemented on commercial digital signal processing hardware. The system achieved high throughput rates, processing more than ten digits per second from the camera input to the classified output.