subject: Signature Verification using a "Siamese" Time Delay Neural Network
Work Project No. 830156-5601
File Case 38543

date: September 2, 1992
from: Jane Bromley
Dept. 11359
HO 4G-338; 908-949-8901
jbrmley@neural.att.com

James W. Bentz
NCR Corp., E&M Atlanta
2651 Satellite Blvd.,
Duluth, GA 30136

Léon Bottou
Dept. 11359
HO 4G-312; 908-949-8901
leonb@neural.att.com

Isabelle Guyon
Dept. 11359
HO 4G-324; 908-949-3220
isabelle@neural.att.com

Yann LeCun
Dept. 11359
HO 4G-332; 908-949-4038
yann@neural.att.com

Cliff Moore
NCR Corp., E&M Atlanta
2651 Satellite Blvd.,
Duluth, GA 30136
cliff.moore@atlantaga.ncr.com

Eduard Säckinger
Dept. 11359
HO 4G-338; 908-949-8901
edi@neural.att.com

Roopak Shah
Dept. 11359
HO 4G-305A; 908-949-9536
rshah@neural.att.com
11359.920902-25TM

TECHNICAL MEMORANDUM

AT&T - PROPRIETARY
Use pursuant to Company instructions
Signature Verification

This is a report on a collaborative project between The Advanced Product Development Department, NCR and The Adaptive Systems Research Department, AT&T Bell Laboratories. The aim of this project was to develop a signature verification algorithm based on the NCR 5990 Signature Capture Device, and using 80 bytes or less for signature feature storage. The 5990 unit provides positional coordinates as a function of time for the pen as it moves over the unit's writing surface.

The signature verification algorithm is based on an artificial neural network. The novel network presented here, called a "Siamese" time delay neural network, consists of two identical networks joined at their output. During training the network learns to measure the similarity between pairs of signatures. When used for verification, only one half of the Siamese network is evaluated. The output of this half network is the feature vector for the input signature. Verification consists of comparing this feature vector with a stored feature vector for the signer. Signatures closer than a chosen threshold to this stored representation are accepted, all other signatures are rejected as forgeries.

As a result of this project, a demonstration system incorporating the neural network signature verification algorithm was developed. This has been shown to commercial customers and a major British Bank is currently planning a field trial in order to examine the feasibility of using this technology.

1 The Aim of the Project

NCR is developing signature capture devices for retail Point-of-Sale terminals. These devices automate credit card transactions by allowing electronic capture, storage and retrieval of "credit receipts of charge" i.e. instead of the currently used "credit card slips" and carbon copies, NCR plans to market a system in which the credit card user signs on a touch-sensitive screen and information about the transaction such as the amount of the purchase and the person's signature are kept in, and printed from, an electronic file. As well as automating credit card transactions, NCR also wishes to provide an extra feature — automatic signature verification. When a signature is electronically captured it can also be automatically verified. Banks currently suffer large losses due to credit card fraud and a major share of these losses is due to lost and stolen credit cards. Signature verification can provide a method to check on the identity of the signer and hence help to reduce these losses.

The aim of the project was to make a signature verification system based on the NCR 5990 Signature Capture Device using 80 bytes or less for signature feature storage. In this TM we report on a method based on artificial neural network technology. Statistical methods have also been investigated by departments 11228 and 11359, and these are discussed in current[8] and following Bell Laboratories Technical Memoranda. Verification using a digitizer such as the 5990, which generates spatial coordinates as a function of time, is known as dynamic verification.

1.1 NCR Requirements

Signature verification works by comparing a sample signature signed on a 5990 Signature Capture Device with a model of that person's signature. This model may be stored in a local file, on a credit card or SMART card, or in a central database.

NCR specified a number of requirements that the signature verification algorithm must fulfill:

- 99.5% of genuine signatures must be accepted while detecting 80% of forged signatures.
- The algorithm should be tunable so that the ratio of the percentage of genuines accepted to forgeries detected can be raised or lowered.

AT&T - PROPRIETARY
Use pursuant to Company Instructions
Signature Verification

- The model of a person’s signature should be about 80 bytes in size.
- The model should become a more accurate representation of a person’s signature with each successful use of the card.

NCR has further requirements for the final product such as it should also be capable of using static (i.e., signatures specified with just spatial coordinates as opposed to dynamic, which also includes timing information) images of signatures, but with less accuracy than that quoted above. It should take less than one second to verify a signature. It should be written in C and use the 80C196 microcontroller running at 12 MHz in the 5990. It should occupy less than 64 Kbytes of read-only and 16 Kbytes of read-write memory and cost no more than $10 of extra hardware. These requirements that can only be addressed once a suitable verification algorithm has been developed and they are not discussed in this report.

2 Data

All signature data was collected using 5990 Signature Capture Devices. The 5990 provides the trajectory of the signature in Cartesian coordinates as a function of time. It also reports whether the pen was touching the writing screen or was in the air, hence information is provided, not only about the motion of the pen on the touch-sensitive screen, but also for the pen in the air (within a certain proximity of the pad). Forgers usually copy the shape of a signature. Using such a touch-sensitive pad for signature entry means that a forger must also copy dynamic information and the trajectory of the pen in the air. Neither of these are easily available to a forger and it is hoped that capturing such information from signatures will make the task of a forger much harder. The spatial resolution of signatures from the 5990 is about 300 dots per inch and the time resolution 200 samples per second (communicated by Jim Benitz, NCR). During the course of this study the 5990 was superseded by the 5991. This device, has a number of extra features over the 5990, but it is also reported to have a much lower spatial resolution of about 100 dots per inch. For this project the majority of the work was carried out using the signature data at 300 dots per inch resolution. However, performance tests were carried out using data treated to have this lower resolution in order to estimate what performance could be expected on the new hardware.

2.1 Data Sets

Two separate data collection efforts were made. The first set (SET1) of 1,636 signatures was collected by Bell Labs and NCR researchers at cafeterias in Bell Labs, Holmdel, N. J. and NCR, Cambridge, Ohio. The second set (SET2) of 1,719 signatures was collected by students at Wright State University. Both sets included genuine signatures and forgeries of these genuine signatures. When producing forgeries, the signer was shown an example of the genuine signature on a computer screen. The amount of effort put into producing forgeries varied. Some people practiced or signed the signature of people they knew, others made little effort. Hence, the quality of the forgeries varied from undetectable to obviously different. Skilled forgeries are the most difficult to detect, but in real life a range of forgeries occur from skilled ones to the signatures of the forger themselves. (Much fraud occurs with credit cards intercepted in the mail. In this case forgers do not have information on the signature of the owner of the card and are forced to invent a signature—often they use their own.) Our database consists of reasonably accurate forgeries, however by using genuine signatures from people other than the original signer we can also measure performance for these so-called zero-effort forgeries.

The data was found to be quite noisy, so Cliff Moore and Sam Wagner of NCR cleaned up the data according to the following rules

\[ \text{AT&T - PROPRIETARY} \]

Use pursuant to Company Instructions
Signature Verification

- Genuine signatures must have between 80% and 120% of the strokes of the signature signed and be of the same name as that typed into the data collection system. This was to remove signatures for which only some part of the signature was present or where people had signed another name e.g. Mickey Mouse.

- Forgeries must be an attempt to copy the genuine signature. This was to remove examples where people had signed completely different names. They must also have 80% to 120% of the strokes of the signature.

3 Neural Network Architectures

Very good performance for dynamic signature verification, well past the requirements set out for this project, can be achieved if no constraints are place on the verification algorithm. Plamondon and Lorette [2] provide a review of current achievements in automatic signature verification. However, once limitations are set, the task becomes much harder. Particularly restricting is the requirement that the model of a person's signature fit within 80 bytes. A person typically takes around 4 seconds to sign their name. For a signature written on the 5990 this gives 800 sets of (x,y and pen up-down (pud) points. In order for a person's signature to be represented in 80 bytes a 30-fold compression must be made, while retaining features of the signature that allow it to be discriminated from forgeries. This need to compress a signature and the observation that with practice, humans can learn to discriminate forgeries from genuine signatures, led us to examine using artificial neural network techniques. Neural networks can learn from example and they can also be used to compress data. The verification process decides on the claimed identity of an author by a comparison process. The work in this report is based on a neural network architecture — the Siamese network — described by Sackinger and Bromley [3] to first extract features from two signatures and then to evaluate the distance between these two sets of features. A similar architecture was independently proposed for fingerprint identification[4]. Chapter 1 in "Introduction to the Theory on Neural Computation" by Hertz, Krogh and Palmer [5] provides a clear introduction to neural networks and we follow their terminology in this report.

The network has two input fields to compare the two patterns and one output whose state value corresponds to the distance between the two patterns. Two separate sub-networks, based on Time Delay Neural Networks (TDNN)[6] [7] act, one on each input pattern, to extract features, then the cosine of the angle between the two feature vectors is calculated and this represents the distance value. The sub-networks are constrained to be identical, they are multi-layer and feed-forward with several layers of feature extraction before the distance measure. TDNN's use processing units having inputs that are delayed in time. In other words, a unit receives inputs from a portion of the signature trajectory. In a multi-layer TDNN, successive layers of units can extract features spread over a wider range of time.

The connections obey the following rules:

- processing units have fields of view which are limited in the time direction,
- the outputs of different units which are members of the same field of view are connected to one feature vector in the next layer,
- the same set of units is repeated along the time axis: the connections between a field of view and a feature vector in the next layer and the connection between the next field of view and the next input vector share the same weights (this operation can be viewed as a nonlinear convolution over the input field).

AT&T - PROPRIETARY

Use pursuant to Company Instructions
This last rule is important when implementing the network – although it may have many thousands of connections this constrains the number of independent weights and hence the memory requirements.

With suitable averaging or sub-sampling the time extent of the network can be reduced. Two network architectures have been investigated, they are depicted in Figs. 1 and 2.

Architecture 1 has 5,347 units, 210,272 connections, but due to the convolutional architecture, only 3,004 independent weights. Architecture 2 has 6,883 unit, 194,216 connections and 654 independent weights. Architecture 1 uses a sub-sampling step of 3, while architecture 2 uses averaging.

When the network is used in the forward direction, the x, y coordinates, pen up/down and a number of other precomputed features (see next section for a discussion of input features) in 200 time steps, are placed on the input to the network. Each processing unit sums its weighted inputs and passes this value through a transfer function to the units of the next layer. The transfer function used is a sigmoid function,

\[ f(x) = \frac{\tanh(2x/3)}{\tanh 2/3} \]

where \( f(x) \) is in the range \(-1\,7159\) to \(1\,7159\), passes through the origin and \( f(1) = 1 \) and \( f(-1) = -1 \).

All the weights of the units are learnt during the training process which uses a modified version of backpropagation\(^8\). To train the network, two patterns are applied one to each input and a desired value for the output is used to backpropagate the error. The desired output is for a small angle between the outputs of the two subnetworks (\( f_1 \) and \( f_2 \)) when two genuine signatures are presented and a large angle if one of the signatures is a forgery. For the cosine distance used here,

\[ (f_1 f_2)/(|f_1||f_2|) \]

the desired outputs were 1.0 for a genuine pair of signatures and \(-0.9\) or \(-1.0\) for the second case. As noted by Säckinger and Bromley\(^9\), there is no clear choice for the desired values. Using 0.0 for the desired output of a genuine-forgery pair gave essentially the same result. The choice of desired outputs should be important for optimal network performance, but it was not a limiting factor on performance in this case and hence, no effort was made to optimize it.

4 Signature Preprocessing

The 5990 provides a representation of a signature as \(x(t), y(t)\) and \(pud(t)\). These values are sampled every 5ms and most signatures are between 300 and 1500 points in length. \(x(t)\) and \(y(t)\) are originally in absolute position coordinates. By calculating the linear estimates for the \(x\) and \(y\) trajectories as a function of time and subtracting this from the original \(x\) and \(y\) values they are converted to a form which is invariant to the position. Then, dividing by the \(y\) standard deviation provides some size normalization. The next preprocessing step is to resample, using linear interpolation, all signatures to be the same length of 200 points. This must be done in order to train the neural network which requires a fixed input size. Next, further features are computed for input to the network and all input values are scaled so that the majority fall between +1 and -1. Ten different features were calculated, but a subset of eight were used as input to the network (see also Fig. 3):

**feature 0** pen up = -1, pen down = +1

**feature 1** \(x\) position, as a difference from the linear estimate for \(x(t)\), normalized using the standard variation of \(y\)

**feature 2** \(y\) position, as a difference from the linear estimate for \(y(t)\), normalized using the standard variation of \(y\)

\[AT&T - PROPRIETARY\]

Use pursuant to Company Instructions
Figure 1: Architecture 1 consists of two identical time delay neural networks. Compression in the time direction is effected by sub-sampling during the convolution.
Figure 2: Architecture 2 consists of two identical time delay neural networks, but only one of them is drawn here. Compression in the time direction is effected by the two averaging layers.
Figure 3: A point on a typical signature trajectory showing the features computed during the preprocessing.

feature 3 speed at each point, spd
feature 4 centripetal acceleration, acc-c
feature 5 tangential acceleration, acc-t
feature 6 the direction cosine of the tangent to the trajectory at each point, \( \cos \theta \)
feature 7 the direction sine of the tangent to the trajectory at each point, \( \sin \theta \)
feature 8 cosine of the local curvature of the trajectory at each point, \( \cos \phi \)
feature 9 sine of the local curvature of the trajectory at each point, \( \sin \phi \)

5 Training the Neural Networks

The data set collected at Bell Labs and NCR, SET1, was used in training. This first round of training was carried out with just part of this data and before it had been cleaned, so it contained noise. Later, a second round of training was carried out using the whole of SET1. By this time the data had also been cleaned. During training, part of the data set is used for learning the weights of the units, another part, called the validation set, is used after a training iteration to test the network’s performance. A measure of the performance of the network at accepting genuine pairs and rejecting forgeries is found by calculating the percentage of genuine signature pairs for which the output (the cosine distance between them) was greater than 0 (genuine accepted, GA) and the percentage of genuine:forgery pairs for which the output was less than 0 (forgeries rejected, FR). During training the network goes through several iterations. The network used for the testing reported in the next
section was the one with the highest GA rate on the validation set. A well-known problem with
backpropagation networks is over-fitting of the data; after much training very good performance
is achieved on the training set, but to the detriment of the network’s generalization ability. By
measuring the network's performance on an independent data set during the training, the version
of the network with the best generalization performance can be found.

5.1 First Round of Training

The first round of training was conducted with a small, noisy data set. The training set consisted
of 5,003 signature pairs; 50% genuine:genreue pairs (g:g), 40% genuine:forgery pairs (g:f) and 10%
genuine:zero-effort pairs (g:0). The zero-effort data was made by pairing genuine signatures of two
different people. The validation set consisted of 1,566 signature pairs in the same proportion as the
training set.

Network 1 Network architecture 1
Input features: pud, acc-c, acc-t, spd, cosθ, sinθ, cosϕ, sinϕ
Best performance on the training set: GA 97.0% FR 65.3% after 26 passes through the training
set
Best performance on the validation set: GA 90.3% FR 74.8% after 6 passes through the train-
ing set.

Network 2 Same as Network 1 except all trajectories when the pen was in the air, were removed
before preprocessing. The aim of this was to investigate whether the pen up trajectories were
too variable to be helpful in verification.
Best performance on the training set: GA 97.8% FR 60.0% after 11 passes through the training
set
Best performance on the validation set: GA 93.2% FR 75.2% after 2 passes through the train-
ing set.

Network 3 Same as 2 except first and last points of pen up trajectories, were included.
Best performance on the training set: GA 99.3% FR 82.6% after 100 passes through the training
set
Best performance on the validation set: GA 93.2% FR 74.6% after 2 passes through the train-
ing set.

Network 4 Same as 1 except x and y were used as input features instead of acc-c and acc-t
Best performance on the training set: GA 99.8% FR 88.8% after 100 passes through the training
set
Best performance on the validation set: GA 91.7% FR 74.2% after 32 passes through the training set.

All these networks show the effect of over-fitting the data. They have learnt the training data by
heart, but their performance on the validation set is poor. This indicates that the training set is too
small. With more data the performance on the validation set will approach that on the training set.

AT&T – PROPRIETARY
Use pursuant to Company Instructions
5.2 Second Round of Training

This round of training used a larger and cleaner training set. The training set consisted of 7,701 signature pairs; 50% genuine-genuine pairs (g:g), 40% genuine-forgery pairs (g:f) and 10% genuine-zero-effort pairs (g:0). The validation set consisted of 960 signature pairs in the same proportion as the training set.

Network 5 Same as 4
Best performance on the training set: GA 98.2% FR 81.7% after 42 passes through the training set
Best performance on the validation set: GA 99.4% FR 80.5% after 42 passes through the training set.
The training simulation crashed after the 42nd iteration and was not restarted. Performance may have improved if training had continued past this point.

Network 6 Same as 5 except architecture 2 was used.
Best performance on the training set: GA 98.6% FR 81.5% after 69 passes through the training set
Best performance on the validation set: GA 99.6% FR 80.1% after 44 passes through the training set.

Network 7 Same as 6 except that the proportion of forgery data to genuine data was increased by presenting the g:f and g:0 pairs four times in every training iteration.
Best performance on the training set: GA 96.1% FR 81.0% after 20 passes through the training set
Best performance on the validation set: GA 98.1% FR 79.5% after 18 passes through the training set.

6 Testing

Two rounds of testing were made. The first round with the unused part of data SET1 and the second with the data set collected at Wright State University, SET2. During testing, signatures to be verified are compared to one single model for the person's signature.

The output values from one half of the Siamese network comprise the signature feature vector for the input signature. We assume that the feature vectors for each person's signatures form a multivariate normal density i.e. that they are continuously valued, mildly corrupted versions of a single prototype vector. For simplicity, we assume that the features are statistically independent, and that each feature has the same variance. During testing, a model is constructed of each person's signature by calculating the mean feature vector and the variance. (See pp.22-27 of "Pattern Classification and Scene Analysis" by Duda and Hart[9] for a fuller description of this.) In the results presented here, the last six signatures signed by each person were used to make the model signature, but any number of signatures could have been used to make the model. In fact the more that are used the better the model will be. With that in mind, the following results should be viewed as a worst case value for performance.

The likelihood that a test signature is genuine, p-yes, is found by evaluating the normal density function. The likelihood of a test signature being a forgery, p-no, is assumed to be a constant value. An estimate for this value was found by averaging the p-yes values for all forgeries. Then the

AT&T – PROPRIETARY
Use pursuant to Company Instructions
Figure 4: Results for Network 1 on the test set (closed circles) and with data simulated to come from the 5991 Signature Capture Device and have 3-fold less resolution in x and y than the standard data (open-circles).

The probability that a test signature is genuine is p-yes/(p-yes + p-no). All results are plotted showing how the percentage of genuine signatures accepted and forgeries detected vary as the threshold probability is varied.

6.1 First Round

This test set consisted of 63 genuine signatures and 63 forgeries for 18 different people. There were about 4 genuine test signatures for each of the 18 people, and 10 forgeries for 6 of these people.

Fig. 4 shows the results for Network 1 tested with the standard data and with the data simulated to have low resolution. The results are essentially identical showing that the verification algorithm developed for the 5990 data can be easily ported to the 5991 environment.

Fig. 5 shows the results for Networks 2 and 3, trained with little or no pen up points. For comparison the result for Network 1 on the standard test set is also shown. The graph shows that there may be some advantage to using just the first and last points as, at least in some part of the range, Network 3 has the best results. However, with such a small test set, this difference may be hardly significant.

Fig. 6 shows the results for Network 4 and for comparison the results for Network 1. This graph shows that using x and y as input features, rather than acc-c and acc-t, has improved performance in some part of the range.
6.2 The Wright State Data

This test set consisted of 532 genuine and 424 forgeries for 43 different people. There were about 12 genuine test signatures for each person, and 30 forgeries for 14 of the people.

Fig. 6 shows the results for Network 4 on this large test set, compared with performance on the smaller test set, closed circles. This result is on a much larger test set and must thus be considered a more accurate measure of the performance of Network 4. Networks were only trained on SET1 and the Wright State data (SET2) was solely used for testing.

Fig. 7 shows the results for Networks 5, 6 and 7. Network 5 has the best performance, closely followed by that of Network 6. Hence reducing by half the size of the model has had little effect on performance. Increasing the proportion of forgeries in the training data has reduced the performance of Network 7.

6.3 Zero-Effort Forgeries

This test again used the Wright State data. The set consisted of 532 genuine signatures, as before, and 430 zero-effort forgeries. For each person, the first genuine signature of the following ten people in the test set was used to make the ten zero-effort forgeries.

Fig 8 shows the results for Network 5 on genuine and zero-effort forgeries. For comparison, the results of testing the same network on genuine signatures and skilled forgeries is also plotted. Surprisingly, Network 5 is better at detecting skilled than zero-effort forgeries. Removing people with very variable signatures from the test, improved the performance on zero-effort forgeries.
Figure 6: Results for Network 1 on the standard test set (open circles) and Network 4 (closed circles) for which x and y were substituted as input features instead of acc-c and acc-t. Network 4 was also tested on the larger Wright State test set (open squares).
Figure 7: Results for Networks 5 (open circles), 6 (closed circles) and 7 (open squares). The training of Network 5 was essentially the same as for Network 4 except that all of SET1 was used in training and the data had been cleaned of noise. Networks 6 and 7 were based on architecture 2. The signature feature vector from this architecture is just 4 by 19 in size. Network 7 was trained on four times as much forgery signature pairs as Network 6.
Figure 8: Results for Network 5 tested on genuine and zero-effort forgeries (open squares), the same test except all people with very variable signatures (variance greater than 0.08) were removed from the test set (closed circles) and for comparison the results when testing with genuines and skilled forgeries (open circle).
6.4 The 80 Byte Constraint

The requirement to represent a model of a signature in 80 bytes means that the signature feature vector must be encodable in 80 bytes. Architecture 2 was specifically designed with this requirement in mind. Its signature feature vector is 4 by 19 in size. When testing Network 6, which was based on this architecture, some outputs were found to be redundant and the signature could be represented by a 38 dimensional vector with no loss of performance. These results were gathered on a Sun SPARC2 workstation where the 38 values were each represented with 4 bytes. A test was made representing each value in one byte. This had no effect on the performance. Using one byte per value allows the signature feature vector to be coded in 38 bytes, which is well within the size constraint. It may be possible to represent a signature feature vector with even less resolution, but this was not investigated. For a model to be updatable, the total of all the squares for each component of the signature feature vectors must also be available. This is another 38 dimensional vector. From these two vectors the variance can be calculated and a test signature verified. These two vectors can be stored in 80 bytes.

6.5 Comments on the Testing

Fig. 9 shows some typical examples of genuine signatures scored with high or low probability of being genuine. It was observed that when a genuine signature was rejected it was very often one of the first written (signatures 8 and 11 of Fig. 9). Signing on the 5990's glass screen has a different "feel" than signing on paper and from our results and people's observations it seems that a person needs some practice at writing on this screen before they can sign consistently. For Network 4 tested with SET2, using a threshold of 0.5 led to 80% of forgeries being detected (as required by the NCR requirements), but only 95.5% of genuines being accepted (24 were rejected). However, if all first and second genuine signatures are removed from the test set the result becomes 97.0% (13 rejected).

Many genuine signatures are low scoring because of uncharacteristic pen up trajectories. In the test of Network 4, once the first and second signatures are excluded 9 of the remaining 13 genuines with low scores had pen up trajectories that differed from the person's typical signature. However, leaving out pen up trajectories does not seem to be the solution to this problem. Network 2 with no pen up information led to lowered performance, Network 3 with very few pen up points gave hardly significant performance over Network 1 which used both pen up and pen down points. The conclusion is that pen up points are useful for some signature verifications and not others. What is required is a method to measure distance between a model and a test signature which allows for flexible matching along the pen trajectory. Suggestions for this are given in the next section.

Due to a quirk in functioning of the digitizer, some parts of a signature that should normally have been pen down, were actually reported as pen up. For the networks which kept both pen up and pen down data, there was some tolerance to this and these genuine signatures could still be correctly verified.

Some people seemed unable to sign consistently and would suddenly miss out letters or add new strokes to their signature. Any verification method has limits; PIN based systems are limited by people forgetting their PIN, signature verification is limited by how consistently people sign. From this trial it seems that, for some people, signing is not a natural and automatically repeatable process. This inconsistency can be compensated for by using many signatures to build a model. However a large variability leads to a large variance for the signature model, which in turn leads to more forgeries being accepted.

Some forgeries were very good. When humans tried to verify them, using just the signature as it would appear on paper, they actually performed worse than the neural networks.

Network 5 was found to be better at detecting skilled forgeries than zero-effort forgeries. There are two explanations for this; the network was trained on very few zero-effort forgeries (5 times

\[\text{AT&T - PROPRIETARY}\]

Use pursuant to Company Instructions
Figure 9: Typical examples of genuine signatures taken from a test of Network 4 on the Wright State data. Left hand column is signatures with high probability of being genuine, right hand column is signatures with low probability of being genuine. The probability value is shown at the bottom right corner of each signature. Both pen up and pen down trajectories are shown.
less zero-effort forgeries than skilled forgeries) and hence never learnt to reliably detect them; secondly, because most forgers are trying to replicate the shape of a signature rather than its temporal characteristics, skilled forgeries tend to have temporal properties uncharacteristic of any signature whereas zero-effort signatures are genuine signatures for someone and hence cannot be detected by an acharacteristic temporal pattern. This also suggests that more training on zero-effort forgeries is required in order for them to be detected reliably.

Architecture 1 has two defects; (i) it performs a compression in the time extent of signatures by using a step size of three for the two convolutional layers and (ii) its output of 16 by 19 units is probably too large to be represented in 80 bytes. Architecture 2 was designed to address these two defects. It does not subsample during the convolutional layers, but instead an averaging layer is introduced after each of these convolutional layers to effect the required compression in time. This method should be more robust to noise in the data. The output size has been reduced to 4 by 19 Performance of Network 6, which is based on the second architecture, was only slightly worse than the best performance of any network using the first architecture. Less effort was spent on fine-tuning this architecture and its training.

The results in this paper should be seen as a lower estimate of possible performance in the real world. The data collected for this task is not a good representation of data from the real world. People could choose whether to sign their own signature or try to forge another from the database and thus does not reflect the ratio of skilled forgeries to genuine signatures that would be expected from a working system. Also, the proportion of forgeries in the real world that may be of the type zero-effort is unknown. There was no incentive for people to sign consistently. In fact, many people deliberately tried to break the system. This led to good forgeries, but also to genuine signers signing inconsistently. As far as was known, no occupational forger signed in our data sets, so we could expect to get better forgeries in a real life system, however, we would also expect more consistency in genuine signatures. All this contributes to making predictions about real-world performance difficult. This question can only be answered by running a field trial. However, results presented here can be compared with other methods which use identical test sets.

7 Future Work

7.1 New Requirements

As work progressed on the signature verification project, NCR amended their list of requirements.—

- Increased accuracy at detecting forgeries of 92% even for the most skilled forger who has knowledge of the signature verification algorithm.

- Be more precise and resistant to forgeries for a person who signs their signature consistently.

- Operate on signatures which have been compressed using a modification of an algorithm developed by Sam Wagner[10].

- Be independent of the general direction in which the signature is signed. i.e. it should work equally well on Western and Arabic signatures.

- It should be insensitive to changes in overall size or of the slope of signatures, and to the addition of a middle initial, but should not accept signatures where the order of first and last name has been reversed, or the signature has been printed rather than written in the normal cursive style or vice versa.

- The algorithm should take no longer than 2 seconds to return a verdict on a signature.
Signature Verification

- It should work with any digitizer that supplies 200 samples per second with resolution 200 dots per inch and pen up and pen down indication.

A number of these new requirements are already met by the method described in this paper. E.g., this method is based on each person's variance in signing their signature which makes it more resistant for people who sign consistently; size and slope variations are removed in the preprocessing; the method is insensitive to the direction of signing. Others, such that the method work for any digitizer, should be easily accommodated with a little work. It is not clear how the requirement that the algorithm work on compressed signatures will affect performance and further work is required to address this. As to the requirement that 92% of forgeries be detected, the current networks do not achieve this and a number of ways to improve this are suggested. There is always a way for a persistent person to beat a system. Once a signature verification system is in everyday use, the deterrent effect may be enough to reduce the general amount of fraud to make the efforts of one determined person inconsequential. However, the ultimate performance can only be measured on a working system.

7.2 Distance Measure using Time Warping

As was mentioned in the previous section, a major cause of false rejection of genuine signatures was signatures which had become distorted in time due to pauses and differing pen up trajectories. By training a variable length input network and then using an elastic matching method for measuring distance, we hope for improvement. This has been shown to lead to improvements on speech recognition using TDNN's[11]. Tools for this are just becoming available and it is strongly recommended that this is investigated. The networks presented here do not achieve the required rejection/accepting accuracy and a distance measure based on time warping offers a solution to this deficit.

7.3 Sign Twice

Another idea which has not been investigated due to lack of time is the increase in accuracy that could be obtained by requiring some or all people to sign twice when being verified. This will certainly improve performance, but at the cost of slowing down the verification process.

8 Conclusions

All except one of the requirements enumerated in Section 1.1 have been met. The algorithm is tunable, a model of a person's signature can easily fit in 80 bytes and the model can be updated and become more accurate with each successful use of the credit card (surely an incentive for people to use their credit card as frequently as possible). A number of the new requirements (Section 7.1) have also been met. The model is more resistant to forgeries for a person who signs their signature consistently, the algorithm is independent of the general direction of signing and is insensitive to changes in size and slope. It was tested at Bell Labs and worked equally well for American, European and Chinese signatures. In its current form, this algorithm does not work on static signatures. Plamondon and Lorette[2] found no automatic verification system for static signatures that could reliably detect skilled forgeries.

NCR wish to store an image of every signature electronically captured. For this reason they introduced the compressed signature format. Compressed signatures include shape and some dynamic information. Depending on how much information is lost in the compression, results for such data...
will approach those for the fully-specified signature. However, this signature format was introduced too late to be included in the study.

As a result of this project, a demonstration system incorporating the neural network signature verification algorithm was developed. This has been shown to commercial customers and a major British Bank is currently planning a field trial in order to examine the feasibility of using this technology.

9 Acknowledgments

All the neural network training and testing was carried out using SN2.6, a neural network simulator package developed by Neuristique. We would like to thank Bernhard Boer, John Denker, Donnie Henderson, Vic Nalwa and the members of Launie Miller's department (11228) at AT&T Bell Laboratories and Sam Wagner at NCR Corporation, for their help and encouragement. Finally, we thank all the people who took time to donate signatures for this project.

Jane Bromley

James W. Bentz ♠

Léon Bottou ♠

Isabelle Guyon

Yann LeCun

Cliff Moore ♠

E. Säckinger

Eduard Säckinger

Roopak Shah ♠

Attn.
References

AT&T - PROPRIETARY
Use pursuant to Company Instructions
Signature Verification

References


