Study of Character Recognition and GUI using Pattern Matching in Neural Networks to improve its Performance

A.Sankaran(P.G Scholar)
Department Of Computer Science Engineering
Indira Institute Of Engineering And Technology

Abstract

Graphical Interface(GI) has been an active area of research in the past and due to its diverse applications it continues to be a challenging research topic. In this paper, we focus especially on offline recognition of handwritten English words by first detecting individual characters. The main approaches for offline handwritten word recognition can be divided into two classes, holistic and segmentation based. The holistic approach is used in recognition of limited size vocabulary where global features extracted from the entire word image are considered. As the size of the vocabulary increases, the complexity of holistic based algorithms also increases and correspondingly the recognition rate decreases rapidly. In this paper we are using the character recognition using pattern matching in a grid box. Which compares the image of alphabets sketched in the grid box, then it searches for the matching alphabets in radio button by using 1 and -1. Which is identified by neural networks(NNs). That how our brain identifies using neurons, dendrites etc for recognition by using a model of "sigmoid" neuron in a neural network.

Introduction

Peripherally reading and talking with colleagues occasionally about the subject for years, found no closer to understanding them than when first learned of them. There has been so much type about the subject of neural nets (NNs). There have also been many attempts to explain how they work found difficult to grasp. The NN concept was the brainchild of mathematicians, and it's no surprise then that people introduce the subject from the perspective of the mathematics. Honestly, though, the nomenclature of the mathematicians' explanations read them is dizzying, despite education.

Obviously without an expert in NNs, able to do with just a little code. Like most NN researchers, In this paper to simulate an NN in software rather than Create one in hardware. It occurred rudimentary experiment might be a good starting point for other programmers curious about the topic. Also, in this hopes that outsider's context might be helpful for programmers and nonprogrammers alike.

Let be quick to disclaim that all here is surely subject to scrutiny. I'm no expert, and some of my explanations and extrapolations may be just plain wrong, so please take them with a grain a salt.

What is a Neural Network?

Without going into detail about the history, let's just say neural networks were invented in the sixties by mathematicians (back in those days, computer scientists were often mathematicians; my how things have changed) looking for ways to model processing in the human brain at the lowest level. They took what they knew about a human neuron as a model for Creating an electronic neuron.

The Neuron

The neural networks approach which currently being used as robot brain only focus on the surface of human brain neurons. Due to that, the ability of autonomous robot to be intelligent as human brain is far from perfection. In order to overcome the weakness of neural networks, researchers should explore the dendrites instead of neurons, because the number of dendrites inside human brain is more than the number of neurons. But the real question is what is dendrites and how to produce the dendritic computations?

These researchers started by boiling down to how they viewed a neuron as working in light of existing knowledge. A neuron, they said, may have many inputs ("dendrites") that a hooked into the single output ("axon") of another neuron.

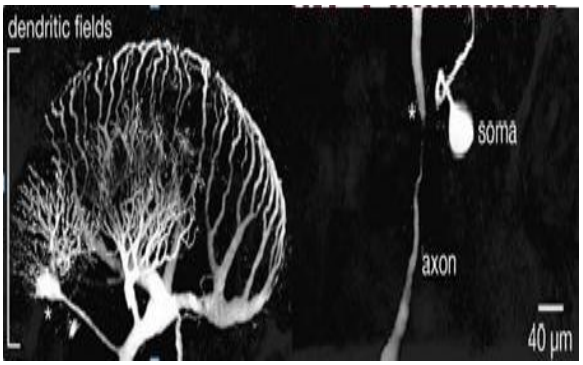

Figure 1 Neuron's dendritic tree

Signals received by some dendrites would tend to activate the neuron, while signals on others would tend to suppress activation. Figure 2 below shows one kind of model of this:

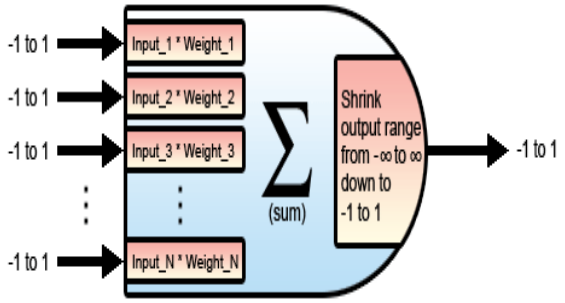

Figure 2: Model of a "sigmoid" neuron in a neural network.

Many essays on how neural networks work are available online. Its difficult to follow; for non mathematician. So by skipping the proofs, then, and summarize the exact algorithm used own neurons for the "thinking" done by a single neuron:

```
Public Sub Think()
  Dim Sum As Single, D As Dendrite
   For Each D In Dendrites
     D.Value=D.FromNeuron.AxonValue
     Sum += D.Value * D.Weight
   Next
  AxonValue=-2/(1+Math.Exp(2*Sum))+1
 End Sub
```

Left some extra lines of code that are to deal with odd cases that are not essential for this discussion. Then variety of ways of Creating a neuron. Some use Boolean values for inputs and outputs, while others use linear (floating point) values. Some range their values from 0 to 1, others from -1 to 1, and still others from -infinity to infinity. The model I chose is referred to as having a "sigmoid" output function. This means that it uses linear input values that range from -1 to 1 and whose outputs, which after summation could be anywhere from -infinity to infinity, are instead "crunched" down to fit in a range from -1 to 1. They are not truncated, but are squashed using the equation on the last line of the .Think() function above.

Pattern Matching

Here is where a lot of introductions stop short. What it means for one of these neurons to "know" something before explaining how they work in concert with other such neurons.

The goal of a neuron of this sort is to fire when it recognizes a known pattern of inputs. Let's say for example that the inputs come from a black and white image 10 x 10 pixels in size. That would mean we have 100 input values. For each pixel that is white, we'll say the input on its corresponding dendrite has a value of -1. Conversely, for each black pixel, we have a value of 1. Let's say our goal is to get this neuron to fire when it sees a letter "A" in the image, but not when it sees any other pattern. Figure 3 below illustrates this:

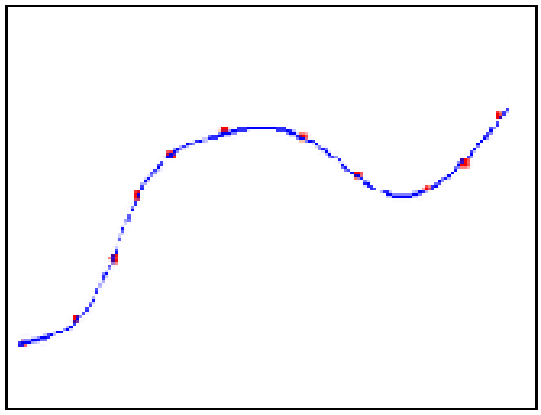

Figure 3: Input values for the dendrites of our neuron.

"Firing" occurs when the output is above some threshold. We could choose 0 as the threshold, but in my program, I found 0.5 is a good threshold. Remember; our only output (axon) for this neuron is always going to have a value between -1 and 1. If 50% of the pixels were "wrong" - not matching what the dendrites say they are expecting as input - the sum would be exactly 0 and hence the output would be 0. Conversely, if the entire image of the "A" were inverted, 100% of them would be wrong, the sum would be -100, and the output would be -1.

Since 0 and -1 are obviously below our threshold of 0.5, we would say this neuron does not "fire" in these cases.

The most important lesson to take away from this simple illustration is that knowledge is encoded in the weights on the dendrites. The second most important lesson is that the output of this sort of thinking is "fuzzy". That is, our neuron compares input patterns and generates variable outputs that are higher the closer the inputs are to its archetypical pattern. So while there are definitive "match" (1) and "non-match" (-1) outputs, there is a whole range in between of somewhat matching.

Training

One might argue that a neuron as desGIibed above can be quite useful as it is for fuzzy matching of patterns, but a little summation and threshold trickery isn't enough to qualify a program as a neural network. One key concept is that a NN can be "trained". In our example above, we might assume that some programmer hard-coded the "knowledge" - the weights for each dendrite - into the neuron at design time. What makes NNs interesting is that they can be given examples of patterns and hence learn to recognize them thereafter, without any designer explicitly calculating what the weights should be. Most explanations of NNs speak of learning as a collective function of a network of neurons, but in this idea of learning in a single neuron instead it's easier to grasp that way.

1. The Categories of Neural Network Learning Rules

There are many types of Neural Network Learning Rules, they fall into two broad categories: supervised learning, and unsupervised learning. Block diagrams of the learning types are illustrated in Figures (1.1) and Figure(1.2).

1.1 Supervised Learning

The learning rule is provided with a set of examples (the training set) of proper network behavior:

{x1, d1} , {x2, d2} , …, {xn, dn} … (1)

where (xn) is an input to the network, (dn) is the corresponding correct target (desired) output. As the inputs are applied to the network, the network outputs are compared with the targets. The learning rule is then used to adjust the weights and the biases of the network in order to move the network outputs closer to the targets (desired). In supervised learning we assume that at each instant of time when the input is applied, the desired response (d) of the system is provided by the teacher. This is illustrated in figure (1), the distance between the actual and the desired response serves as an error measure and is used to correct network parameter externally.

For instance, in learning classifications of input patterns or situations with known responses, the error can be used to modify the weights so that the error decreases. This mode of learning is very pervasive. Also it is used in many situations of natural learning. A set of input and output patterns called training set is required for this learning mode .

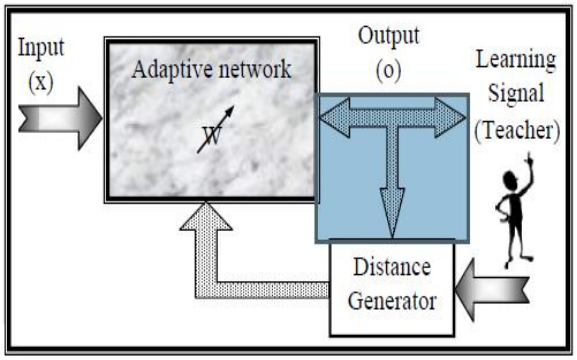

Figure 1.1: Block diagram for explaining of supervised learning.

1.2 Unsupervised Learning

In unsupervised learning, the weights and biases are modified in response to network input only. There are no target outputs available. At first glance this might seem to be impractical. How can you train a network if you don't know what is supposed to do? Most of these algorithms perform some kind of clustering operation. They learn to categorize the input patterns into a finite number of classes. This is useful in such applications such as vector quantization . Figure (1.2) shows the block diagram of unsupervised learning rule. In learning without supervision the desired response is not known; thus, explicit error information cannot be used to improve network behavior.

Since no information is available as to correctness or incorrectness of responses, learning must somehow be accomplished based on observations of responses to inputs that we have marginal or no knowledge about.

Unsupervised learning algorithms use patterns that are typically redundant raw data having no label regarding their class membership, or associations. In this mode of learning, a network must discover for itself any possibly existing patterns, regularities, separating properties, etc., while discovering these the network undergoes change in its parameters, unsupervised learning is sometimes called learning without teacher. This terminology is not the most appropriate because learning without a teacher is not possible at all. Although, the teacher does not have to be involved in every training step, he has to set goals even in an unsupervised learning mode.
Learning with feedback, either from the teacher or from environment, however, is more typical for neural network. Such learning is called incremental and is usually performed in steps. The concept of feedback plays a central role in learning.

The concept is highly elusive and somewhat paradoxical. In a broad sense it can be understood as an introduction of a pattern of relationships into the cause-and-effect path.

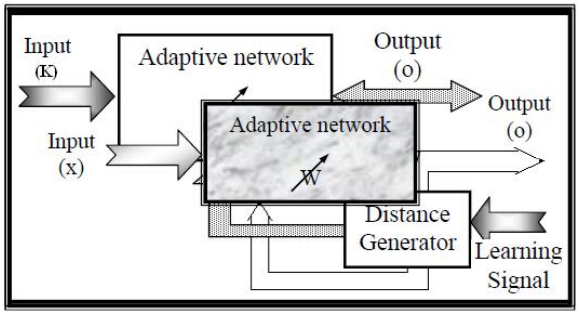

Figure (1.2) Block diagram for explaining of unsupervised learning.

1.2. Neural Network Learning Rules

Our focus in this section will be on artificial neural network learning rules. A neuron is considered to be an adaptive element. Its weights are modifiable depending on the input signal it receives, its output value, and the associated teacher response. In some cases the teacher signal is not available and no error information can be used, thus a neuron will modify its weights, based only on the input and / or output. This is the case for unsupervised learning.

The trained network is show in Figure (3), It study the weight vector (wi) or its component (wij), which connecting the (j'th) input with neuron (i). The output of another neurons can be the (j'th) input to the neuron (i).
Our discussion in this section will cover single neuron and single layer network supervised learning and simple cases of unsupervised learning. The form of neuron activation function may be different when different learning rules is considered. The learning ability of human beings is properly incorporated in the facility of the changing the transmission efficiency of the synapses which corresponding to adaptation of the weight. The convergence has always been a major problem in the neural network learning algorithms.

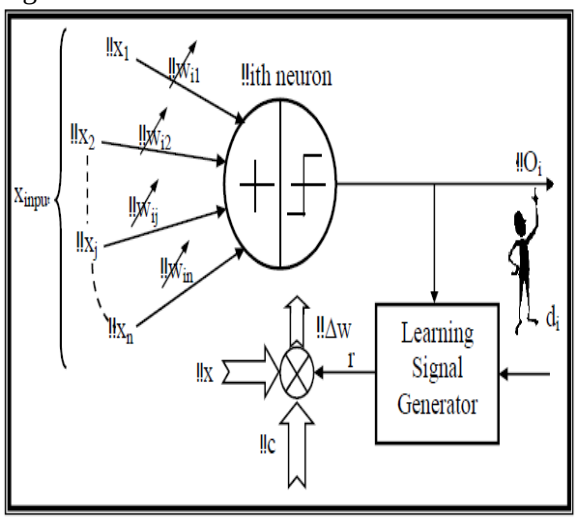

Figure (1.3) Illustration of weight learning values (d_i) provided only for supervised learning mode)

In most cases, to avoid this, impractical applications different initial conditions are used until one case would converge to the desirable target. The weight vector wi=[wi1 wi2 wi3 …. win] t increases in proportion to the product of input (x) and learning signal (r). The learning signal (r) is, in general, a function of (wi,x), and sometimes of the teacher's signal (di). So for the network shown in Figure (3.3):

$$r = r(wi,x,di) \ldots (2)$$

The increment of the weight vector (wi) product by the learning step at time (t) according to the general learning rule is given by

$$wi(t)= c\, r[wi(t), x(t), di(t)]\, x(t) \ldots (3)$$

where (c) is constant called the learning constant that determines the rate of learning. At the next instant learning step, the weight vector adapt at time (t) becomes:

$$wi(t+1)= wi(t) + c\, r[wi(t), x(t), di(t)]\, x(t) \ldots (4)$$

1.3 Hebbian Learning Rule

For the Hebbian learning rule the learning is equal simply to the neuron's output as shown in Figure (3.4), so it can be seen that:

$$r = f(wtix) \ldots (5)$$

The increment (wi)of the weight vector becomes:

The resulting set of weights will be a compromise that equally balances the interests of all the training cases. Following is the actual code for a neuron to execute one training cycle:

```
Public Sub AutoTrain()
    Dim T As TrainingCase, Td As
TrainingCaseDendrite
    Dim ErrorTerm As Single
    Const LearningRate As Single = 0.01

    'Learn from each training case in this
one cycle
    For Each T In Me.TrainingCases

        'Preset my dendrite's source axons'
values to reflect
        For Each Td In T.Dendrites

Td.ForDendrite.FromNeuron.AxonValue =
Td.Value
        Next

        'Given the training inputs, generate
an output
        Think()

        'Morph toward the desired output
        ErrorTerm = T.AxonValue -
Me.AxonValue
        For Each Td In T.Dendrites
            Td.ForDendrite.Weight +=
Td.Value * ErrorTerm * LearningRate
        Next
    Next
End Sub
```

After a few dozen invocations of this routine for this neuron, the ability to match even a sloppily drawn letter is surprisingly good. Sample program shows the matches as checkboxes that suggest definitiveness, but in fact every letter's neuron has its own output from -1 to 1 indicating how closely it believes the input value matches its view of how such a letter should appear.

The Network

The explanation above gives a sense of the power of an individual neuron to recognize and even "learn" to recognize patterns and to fire (or not) based on recognizing such known patterns. We get into the heart of neural networks' power when we start to link them together.

But it seems in practice, there are usually only two or three.

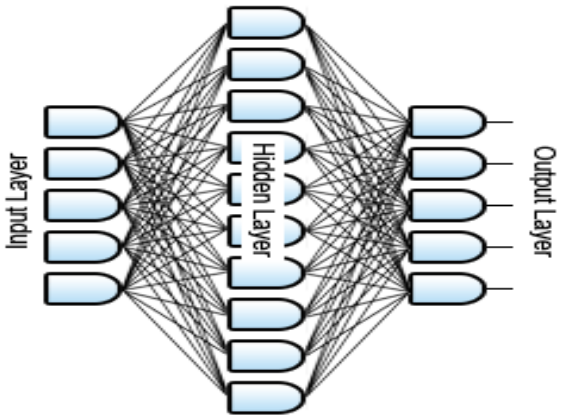

Figure 4: Schematic of a neural network with three layers of neurons.

The first layer, referred to generally as the "input" layer, is actually just there to make programming easier. Its neurons don't actually have any dendrites. Instead, the input of some input matrix - say, an image of a letter - gets pumped straight into the axon values for each neuron. The neurons themselves are just placeholders so the next layer can tap into these input values in the same way each subsequent layer does.

If two layers is good enough for recognizing a pattern, then you might well ask what the point of adding another layer is. The point, generally, is to add layers of abstraction. Let's take the Graphical Interface example and say that the neurons that recognize characters are now in the hidden layer and we add a new output layer. Perhaps our goal is to determine when a character recognized represents an even or odd character, in an ordinal sense. The training cases for the "even" neuron would indicate that the neuron should fire when the previous layer fires to indicate one of the even

characters and should suppress any urge to fire when it indicates one of the odd characters.

Biological Neural Nets

In biological neural net, neurons are living cells with axons and dendrites that form interconnections through electro-chemical synapses. Signals are transmitted through the cell body (soma), from the dendrite to the axon as an electrical impulse. In the pre-synaptic membrane of the axon, the electrical signal is converted into a chemical signal in the form of various neurotransmitters. These neurotransmitters, along with other chemicals present in the synapse form the message that is received by the post-synaptic membrane of the dendrite of the next cell, which in turn is converted to an electrical signal.

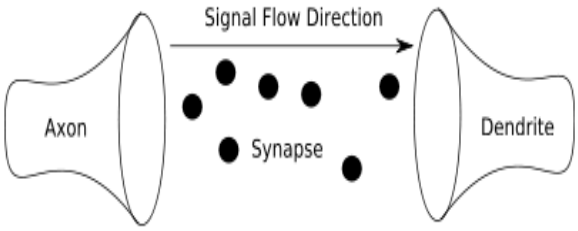

Figure 5: model of the synapse chemical messages of the synapse moving from the axon to the dendrite.

The above figure 5 shows a model of the synapse showing the chemical messages of the synapse moving from the axon to the dendrite. Synapses are not simply a transmission medium for chemical signals, however. A synapse is capable of modifying itself based on the signal traffic that it receives. In this way, a synapse is able to "learn" from its past activity. This learning happens through the strengthening or weakening of the connection. External factors can also affect the chemical properties of the synapse, including body chemistry and medication.

Not Like Our Brain

There has been a bit of a mystique and aura surrounding them. They are often sold in the popular press as pared down electronic versions of the way the human brain works. They are also envisioned by some as the panacea to the artificial intelligence problem. I suspect that serious researchers in this field know better.

Let a sigmoid neuron of the sort GI is not a bad model for how an individual neuron in the human brain functions; I'd be willing to buy that. But no doubt that neurons in the brain are not lined up in layers illustrated in figure 4 above, with each layer serving a different functional purpose and each neuron only talking to neurons in adjacent layers. Evolution is just not that good of a planner. With each new phase of experimentation, evolution makes subtle adjustments that serve some short-term good of a given individual. They are always "band aid" solutions. The result is a very function but hideously complicated tangle of neurons.

Second, neurons in the brain are not all the same. Some have short axons and dendrites that only span perhaps millimeters and thus propagate local information, while others can have axons that run many inches or perhaps even several feet. The neurons in your spinal column are structurally and functionally different from the ones in your brain. The ones in your brain vary by where they are, and they all certainly have differing functions. Most artificial neural nets, by contrast, are composed of just one type of neuron, and they are all in a very homogeneous kind of arrangement.

Third, the ideas of training cases and back-propagation seem intuitive enough as a learning paradigm for NNs, but I've never seen any indication that such mechanisms exist in the human brain. Neurons don't take time off every now and then to undergo training and then get switched back into a performance mode. They seem to learn and perform as they go.

So I think I can fairly say that, no matter how many layers deep and how many neurons wide one makes a neural network, no such network will ever be able to function like a human brain or even become vaguely sentient. That said, I think it's fair to also say that NNs can play a role in AI research. Perhaps an intelligent being can be borne out of an assemblage of just NNs, but it can't just be one big stack. It would have to be lots of individual stacks- what I call a "cluster" of stacks - interacting. Most of the parts of the cluster would have to be pre-trained so they can perform largely unchanging, autonomous functions.

Conclusion

In this paper able to function in such a way that learning and acting can happen in parallel, or at least in alternating sequences.

More realistically, NNs can be used as just one kind of machine in a heterogeneous platform of technologies to provide practical value. Perhaps one part of an AI might decide it wishes to learn to recognize certain patterns in its vision - perhaps different faces or machine parts - and allocate a new neural-net to do the job and control when and what it learns, for example.

It seems the science of neural networks is largely dead and NNs have settled into being an esoteric, if rarely-used, tool in various practical applications like cameras and manufacturing processes.

References

[1]Yi-Chong,Zeng Inst. of Inf. Sci., Acad. Sinica,Taipei,Taiwan, "**Pattern recognition using rotation-invariant filter-driven template matching**," Image Processing (ICIP), *11-14 Sept. 2011 18th IEEE International Conference*, pp. 2389 - 2392 .

[2]Vishwaas, Arjun, M.M. ; Dinesh, R. JSSATE, Bangalore, India "**Handwritten Kannada character recognition based on Kohonen Neural Network**," *Image Processing (ICIP), 25-27 April Recent Advances in Computing and Software Systems (RACSS), 2012 International Conference*, pp. 91 - 97 .

[3]Gupta, Srivastava, M. ; Mahanta, C. Dept. of Electron. & Electr. Eng., IIT Guwahati, India "**Offline handwritten character recognition using neural network**," *Computer Applications and Industrial Electronics (ICCAIE), 2011 IEEE International Conference*, pp. 102 - 107 .

[4] Pradeep, Srinivasan, E. ; Himavathi, S. Dept. of ECE, Pondicherry Eng. Coll., Pondicherry, India "**Neural network based handwritten character recognition system without feature extraction**," *Computer, Communication and Electrical Technology (ICCCET), 2011 IEEE International Conference*, pp. 40 - 44 .

[5] Aiquan Yuan , Gang Bai ; Lijing Jiao ; Yajie Liu Coll. of Inf. Tech. Sci., Nankai Univ., Tianjin, China "**Offline handwritten English character recognition based on convolutional neural network**," *27-29 March 2012 Document Analysis Systems (DAS), 2012 10th IAPR International Workshop*, pp. 125 - 129 .

[6] Rashid, S.F.Shafait, F. ; Breuel, T.M. Dept. of Comput. Sci., Tech. Univ. Kaiserslautern, Kaiserslautern, Germany "*27-29 March 2012* **Scanning Neural Network for Text Line Recognition**," *Document Analysis Systems (DAS), 2012 10th IAPR International Workshop*, pp. 105 - 109 .

[7] Shenghui Wang Fei Yang Inf. Technol. Center, China Guangdong Nucl. Power Holding Co. Ltd., **Research in Character Recognition Based on Generalized Regression Neural Network**," *Computing, Measurement, Control and Sensor Network (CMCSN), 2012 International Conference*, pp. 318 - 320 .

[8] Gheorghita, S. Munteanu, R. ; Enache, M. Faculties of Electr. Eng., Tech. Univ. of Cluj-Napoca, Cluj-Napoca, Romania , India "**Study of neural networks to improve performance for character recognition**," *Automation Quality and Testing Robotics (AQTR), 2012 IEEE International Conference*, pp. 323 - 326 .

[9] Nan-Chi Huang, Huei-Yung Lin ,Dept. of Electr. Eng. & Adv. Inst. of Manuf., Nat. Chung Cheng Univ., Chiayi, Taiwan "**A multi-stage processing technique for character recognition**," *Advanced Intelligent Mechatronics (AIM), 2012 IEEE/ASME International Conference on*, pp. 1081 - 1085.