Learning Categories with Spiking Nets and Spike Timing Dependent Plasticity

Christian Huyck[1]

Middlesex University, London NW4 4BT UK
c.huyck@mdx.ac.uk
http://www.cwa.mdx.ac.uk/chris/chrisroot.html

Abstract. An exploratory study of learning a neural network for categorisation shows an effective use of commonly used leaky integrate and fire neurons and Hebbian learning. The system learns with a standard spike timing dependent plasticity Hebbian learning rule. A two layer feed forward topology is used with a presentation mechanism of inputs followed by outputs a simulated millisecond later to learn Iris flower and Breast Cancer Tumour Malignancy categorisers. An exploration of parameters indicates how this mechanism may be applied to other tasks, and extensions to the topology, and overall mechanism are explored.

Keywords: Spiking Neural Network · Spike Timing Dependent Plasticity · Categorisation

1 Introduction

AI is a critical technology with immense interest from governments, companies and society at large. Recent developments in machine learning and deep networks in particular have achieved success in a wide range of areas, such as face recognition [18] and games [16, 13].

Deep Nets [9] are a diverse group of systems typically with large numbers of units between layers, and many well connected layers. Deep nets gain their strength from these connections, which are trained to reflect a mapping from known inputs to known outputs. These connectionist systems are typically inspired by the brain, and are thus called neural networks.

Simulated biological neural networks, on the other hand, attempt to reproduce the behaviour of brains, or parts of brains [15]. These are based on models of biological neurons, models of biological learning, and biological topologies. The neural models are typically spiking neurons.

Research in deep nets and biological neural nets are largely distinct, though there is shared interest. In particular, one of the problems with work in deep nets is that it is computationally expensive, while actual biological nets are known to be relatively computationally inexpensive. A human brain uses less power than a light bulb. Consequently, there is an increasing interest in spiking nets for machine learning.

This paper describes a system that categorises data using a simulated neural network. The network has aspects of biological plausibility combined with a biologically unrealistic topology. The plausible aspects include learning via spike timing dependent plasticity, a Hebbian learning rule, and a widely used, though simple, biological neuron model. It is not clear that the presentation or testing mechanism is psychologically realistic.

2 Literature Review

There many neural models including relatively simple point models that represent neurons by simple equations and elaborate compartmental models [10] that break neurons into compartments and evaluate the electrical flow through these compartments. Leaky integrate and fire neurons integrate activation from other neurons. The activation leaks away, but if enough accumulates, the neuron fires emiting a spike. The activation resets after it fires. In this paper, the leaky integrate and fire neural model is from Brette and Gerstner [5]. The model includes exponential current transmission, so that the current is transferred across the synapse (after the pre synaptic neuron fires) at an exponentially decaying rate.

In the brain, most if not all learning is Hebbian [8]. If the pre synaptic neuron tends to cause the post synaptic neuron to fire, the weight will tend to increase. There are many variations of this rule, but a great deal of biological evidence supports Spike Timing Dependent Plasticity (STDP) [3]. Bi and Poo [3] have perhaps the first published example that shows the performance of the changing efficiency of biological synapses. Song et al. [17] have developed an idealised curve that fits the biological data, though it is a curve fitting exercise. Figure 1 shows a version of this curve. Horizontally, the figure shows the difference between the firing times of the pre and post synaptic neurons, with the post synaptic neuron firing first on the left. In this case, the synapse is weakened because the pre synaptic neuron has not caused the post synaptic neuron to fire. When the pre synaptic neuron fires first, the weight is increased. Note that the closer to precisely co-firing, the more the weight change. These curves are exponential functions. The simulations in the remainder of this paper use an STDP learning rule.

Other spiking systems have used Hebbian learning and Hebbian like learning rules to learn categorisers. One system uses leaky integrate and fire neurons with adaptation and a compensatory Hebbian learning rule to learn categorisers [11]. This uses recurrent connectivity. Another uses spiking neurons and a modified Hebbian learning rule [20]. The modified rule includes data about the slowly varying time-averaged value of post-synaptic activity [4].

3 Methods

There are many neural simulators and neural models. This diversity encourages exploration, but reduces reusability and reproducibility. The code for the simulations below makes use of one widely used mechanism, PyNN and NEST. The

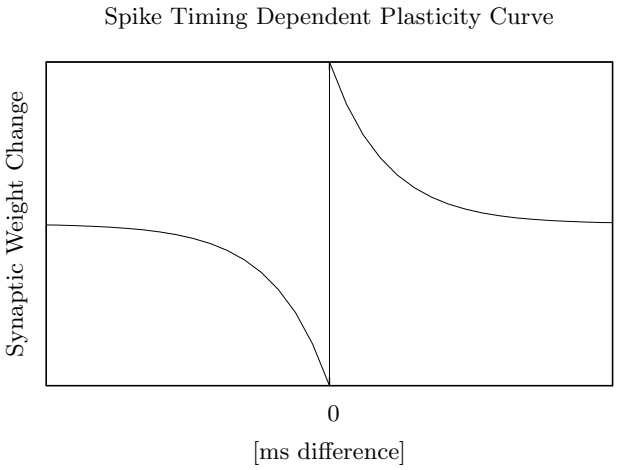

Fig. 1. An idealised standard STDP curve. The left curve shows the weight change of the synapse that connects two neurons when the pre synaptic neuron fires first. It is reduced, and reduced more the closer the two fire. The left curve shows the weight change when the pre synaptic neuron fires first; the weight increases, and the closer the two fire the more it increases.

system was developed using PyNN middleware [6], a python package to specify the neural topology and manage inputs. The backend was the NEST neuron simulation platform [7].[1]

Data was taken from the widely used University of California at Irvine (UCI) benchmark [1]. A commonly used task, categorisation of Iris flowers, is used initially. The data was split into two equal sized groups. The Iris data has 150 instances, 50 of each of three categories, so the data was split into two 75 item data sets with 25 of each category in each.

First the data is preprocessed by scaling the range of features to 0 to 100 with two digits of precision. Now all features are represented by a number between 0 and 100 inclusive. The input to the system is represented by a neuron for each number. So, for the Iris data, there are four features, and thus 404 input neurons.

There is a neuron for each output category. For the Iris data, there are three categories, and thus three neurons.

The input neurons are well connected to the output neurons using plastic synapses. The plasticity rule is a variant Hebbian STDP, consistent with Song et al. [17].

During training, the input neurons are sent a spike, and the output neurons are sent a spike one ms. later. This uses the PyNN spike source, an impossible feature of typical biological learning because neural firing is caused either directly in sensory neurons, or via a cascade of firing from other neurons. It could

[1] The code can be found on http://www.cwa.mdx.ac.uk/NEAL/NEAL.html.

duplicate the behaviour when an electrode is put into a neuron. The input neurons consist of those mapped to the input feature, and in a window of three on either side. So, when the first training feature is 19, the neurons numbered 15 to 21 are stimulated, as numbering is zero based. The neurons are stimulated so that they fire once.

One of the parameters that was explored in development was the number of training epochs. An epoch is the presentation of all the training examples; in the case of the Iris task, this is all 75 training items. There may be several epochs of training with all of the training items presented in sequence in each epoch.

The time between each example was another parameter that was explored. This time can effect the system for three main reasons. First, if the time is too small, an input example can continue to spike into the next example. Second, the STDP synaptic reduction window is effected by the prior example; if the prior example fires nearer to the time of the current example, the synaptic weight from the prior input neurons to the current category neuron may be reduced. Third, the simulated neurons are dynamic, slowly returning to a resting state over time.

During testing, the input neurons are stimulated, and the spikes of the output categorisation neurons are counted. The input is categorised based on the spiking behaviour with the neuron that spiked most winning. If there is a tie it was resolved as follows: no neurons fired, category 3; all three neurons fired equally, category 2; the first and second neuron were co-equal highest firers, 2; the first and third neurons were co-equal highest firers, 3; and the second and third were co-equal highest firers, category 2. Other measurement mechanisms could have been used or included, for example the first spike time could be used as the answer, or to resolve ties.

It is possible to set the neural parameters for the model, but the default parameters were used, and varying them was not explored. Some important parameters are: the firing threshold, the higher the threshold, the more activation is required for a spike; the refractory period, after a neuron spikes, all input activation is ignored during the refractory period; and the leak rate, the higher the leak rate the faster activation leaks away making the neuron more difficult to fire over short periods of time. Another simulation parameter is the time step that sets how often are the neural and synaptic variables are updated. These simulations used a one ms. time step.

There are many STDP rules, and a spike pair rule is used in the simulations in this paper. The STDP learning rule is described by seven parameters. The first is the initial synaptic weight i, the second is the maximum synaptic weight m. The third is the minimum weight that has always been 0 in the simulations described in this paper; input features may have no influence on output categories, so the neurons that represent these values should have a 0 connection. The weight is modified based on spike pairs alone. There are four parameters associated with this, two for increasing the weight and two for decreasing the weight. The increasing parameters are $A+$, for scaling how much the weight increases, and

$\tau+$ for stretching the window that the weight increases. The parallel decreasing parameters are $A-$ and $\tau-$. $\tau+$ and $\tau-$ are in ms.

Parameters are explored to develop a system that categorises reasonably well. The data has been broken into a training and a test set. Parameters are explored using the training set, and the test set is used, largely, for reporting.

In the first example there are five training epochs; the time between examples is 30; the initial synaptic weight was 0; the maximum synaptic weight was 0.05; the synaptic increase parameters were $\tau+ = 20.0ms.$ and $A+ = 0.004$, and the decrease parameters were $\tau- = 20.0ms.$ and $A- = 0.003$. This is represented by the first line in table 1. The results of this system is 65 of the 75 training items correctly categorised when the system was trained on the first set and tested on the second; the result, pleasantly, was 68 of the 75 when the training sets are switched.

4 Results

Exploration of parameters can include a change of topology, but some simple things to explore are the learning parameters, and presentation mechanism. In particular, the five learning parameters, the number of training epochs, and the length between presentations are explored.

Exploration is done by training the system on the training set, and testing on the testing set. It is not a two fold cross validation. Aside from randomising presentation in section 4.1, the system uses no randomness. Repeating a run will have the same results. Fortunately, each run provides information.

One large piece of information is how many test examples actually have neurons firing. The default categorisation works with no neurons firing but it shows that the system is not categorising these examples. The parameters affect how many categorisation neurons fire in the test. Unsurprisingly, typically, the more training epochs, the more firing in the test neurons. Since synaptic weights are initially zero and each presentation provides the opportunity to increase, and increased synaptic weight leads to further firing in future epochs, further increasing weights, more epochs lead to increased firing in the test neurons. This may not be the case if the initial weight is larger than zero (see section 4.1).

Too much firing also causes the output neurons to become saturated. The neurons integrate input from a set of pre synaptic neurons, all firing once at the same time, and this can cause the post synaptic neuron to have enough activation so that it fires multiple times. However, if two or three of the categorisation neurons get a great deal of input, they saturate and fire the same number of times. So, there is an ideal window of incoming synaptic strength to differentiate between the categories.

Perhaps the most powerful mechanism for increasing firing is to increase the maximum synaptic weight m. So, if the system with a particular parameter set had many tests with no output neurons firing, m was increased. Similarly, the total output spikes can also be tabulated, and if this is very high, m can be reduced.

The synaptic weight increase and decrease constants $A+$ and $A-$ also influence output neuron firing. Increasing $A+$ or decreasing $A-$ leads to increased firing, while decreasing $A+$ or increasing $A-$ leads to decreased firing.

The synaptic change windows, $\tau+$ and $\tau-$, also affect the firing. In general the influence is in the direction of their associated weight change constants, but it is not always the case. Also changing the windows has less influence than changing the weight constants. Note that the biological interpretation is that $\tau+ = \tau- = 20.0ms$.

There is also an affect from changing the length of presentation time. During training, the synaptic weight is reduced from a feature to its prior category, and this is reduced less when the time between presentations is greater. Moreover, neurons are dynamic, when they fire they are set to a reset voltage. If they have voltage above or below the resting voltage, that leaks away moving the voltage toward resting. With a longer example time, the neurons have returned closer to their resting state by the beginning of the next presentation.

The goal is to have a system that categorises well. So, the categorisation results also matter. By following the gradient so that all tests have categorisation neurons firing, but many with only one spike, parameters can be set to find a good result on the training set. One such parameter set is shown in the second row of table 1.

Two other systems are shown for comparison. The first [11], in the third row is a spiking net using a compensatory Hebbian rule; the neurons have adaptation. As it uses randomness, the average results of a two fold test are shown. The second [20] uses a specialised feed forward neural topology and a variant of STDP that incorporates a learning signal; training and testing are separated in their evaluation, and in table 1.

Table 1. Categorisation Results: WBC refers to the Wisconsin Breast Cancer task.

Task	Epochs	Example Time	Maximum Weight m	$A+$	$\tau+$	$A-$	$\tau-$	Train Result	Test Result
Iris	5	30	0.005	0.004	20.0	0.003	20.0	86.67%	90.6%
Iris	6	50	0.003	0.005	20.0	0.002	20	92%	90.6%
Iris [11]									93.53%
Iris [20]								95.5%	95.3%
WBC	6	50	0.002	0.006	20.0	0.009	20	95.14%	95.7%
WBC [20]								96.2%	96.7%

4.1 Unsuccessful Extensions

Several other options were explored, though no significant benefit was found for the Iris categorisation task. The first was to use a pairwise feature combination categorisation as part of a three layer topology. This is shown in figure 2.

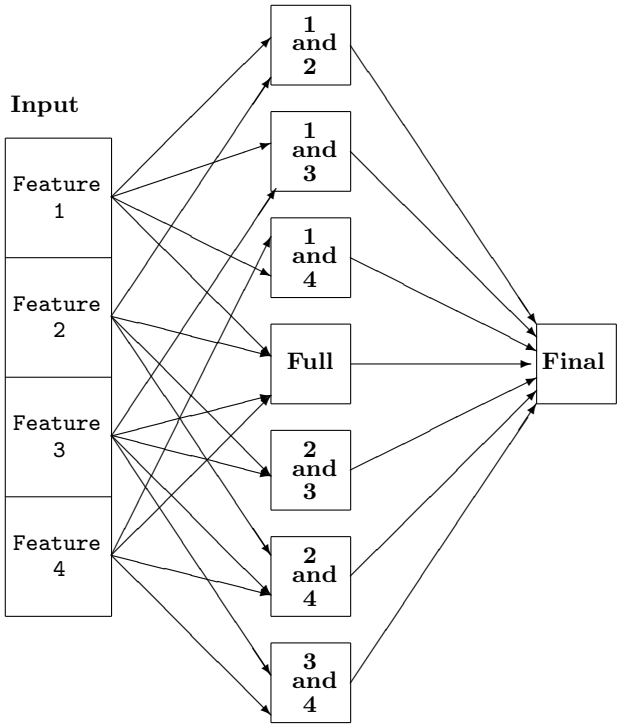

Fig. 2. The three layer feature Iris categorisation topology. The Input box consists of 404 neurons representing four features. The lines from the input boxes to the middle layer represent 303 plastic synapses, since the middle boxes represent three neurons for the categories. The arrows from the middle to the output layer represent nine static synapses. Note that the original system is just the **Input** box and the **Full** box.

In the pairwise combination, each feature pair was also used to learn to categorise the training data. None of these pairs ever did as well as the full system. All nine sets of categorisation neurons, one for each of the six pairs and one for the full set, then influenced the respective output neurons in the firing layer. Different weightings were explored so that some feature pairs had more influence than others, but the best result was always dominated by the full system, and was never better than the full system. This does however provide a mechanism to show how influential pairs are. When exploring a new categorisation domains, different single, pair and higher order combinations can be explored to show that some combinations are particularly influential or not very useful.

Another exploration was a parameter search of initial synaptic weight i. The default value was 0, the value can only be positive, and should be less than the maximum synaptic weight m. A larger value of i does lead to more neural firing, but it seemed to lead to poorer results. Perhaps this was because forgetting was weak, or that higher i put the system in a bad part of the parameter space. Higher i should, however, make the system converge more quickly (see section 5).

Presentation of training data was in the same order each epoch. So, synapses would only be adjusted downward on particular pairings and some might be missed. Changing this to randomising the presentation order each epoch led to an increase of firing. The synaptic weights are adjusted up each presentation, but only down on random presentations. Changing to random presenatation did not lead to a better categoriser, but this warrants more exploration.

These methods have only been explored on Iris categorisation. They may have positive effects on other categorisation tasks, or indeed further exploration may show that these have benefits.

4.2 Wisconsin Breast Cancer Categorisation

A modified system was developed for a second task, the Wisconsin Breast Cancer categorisation task, again from the UCI benchmark [1]. An item refers to a patient and is represented by nine relevant features, and the output is a binary value referring to whether the tumour was benign or malignant. There were 699 items, with 241 malignant category items. The data set was split into two with the training set having an extra item and an extra malignant item.

Each feature had a range from 1 to 10, one feature had missing values, and one feature had one value that was not represented. So, each input feature was represented by 10 neurons and the two output categories by one neuron each. When an item was presented, only one neuron was stimulated for each feature, and the missing feature was simply ignored; one of the benefits of this approach is that missing features are readily ignored.

A simple exploration of the parameter space began with the parameters from the Iris data set (line 2 of table 1). This exhibited a great deal of firing during testing, so the Maximum Weight w was reduced. This left little firing, so $A+$ was increased. Somewhat surprisingly, increasing $A-$ also improved results leading to several training parameter sets that got 95.14%. One was chosen, and the

results are displayed in the first WBC line of table 1. Below that the results reported from another spiking system [20] are shown.

5 Discussion

This paper has shown an exploration of a standard Hebbian STDP learning rule based on a standard biological leaky integrate and fire neuron with a simple well connected feed forward topology. The presentation mechanism of turning on the input neurons one step before the output neurons is not the way brains typically learn. Similarly, the uniformity of the initial feed forward topology is also biologically implausible. While the results are below the state of the art, and the tasks are simple, the results are quite near the state of the art. This shows how powerful the strictly Hebbian STDP learning mechanism is.

STDP, with the topology and presentation mechanism used above, has a result that is a type of covariance mapping. The synaptic weight from a neuron representing an input feature will increase if it is used as a member of the category. If it is also used for another category, it will decrease, so the weight roughly reflects the likelihood the feature discriminates between the categories. If it is involved in two categories, the weight will be lower, and if in three lower still. The feature breadth mechanism used in the Iris task supports learning from fewer examples, and generalisation to unpresented data.

STDP is strictly Hebbian, so is an entirely unsupervised mechanism. Reinforcement can be included by adding extra topology to encourage neurons to fire at appropriate times; this is a solution that has been included in spiking networks [2, 20]. Adjusting synaptic weights to reflect desired outputs, as is done in supervised rules such as back propagation [14] does not seem to have a biological basis. This supervised learning is a powerful mechanism, particularly for feed forward networks. However, the brain is not feed forward but highly recurrent.

This paper has used a simple two layer feed forward approach, though a third layer was explored in section 4.1. Learning here is based on particular inputs. Another approach would be to extend across layers with different times so that input could cascade through layers. For example, the feature approach in section 4.1 could be extended to allow learning between the second and third layer with firing in the third layer specified after the clamped firing in the second. Other precise timing mechanisms can be developed, but it must be remembered that in the brain, most neurons fire more or less continuously at a low rate. Closely timed mechanisms will probably be poor models of actual biological processing.

The learning networks that have been described in this paper are not stable in the sense that further learning will change the weights. Eventually, given enough presentation epochs, the system will become largely stable with weights not changing at the end of each epoch. Weights that are only associated with one category will go to the maximum weight m, those that are not associated with a category will remain 0, and ones that are associated with multiple categories will be lower. These systems have not gotten to this stage, so will change with

further training. Indeed, they are changing during testing. A brief exploration indicates keeping synapses constant during testing does have an effect.

6 Conclusion

This paper has shown that simple well founded neural models with Hebbian rules can be used to learn effective categorisers. These biologically realistic components can be combined in feed forward networks and learn from a simple presentation mechanism. Further exploration of this mechanism may lead to further machine learning improvements. However, it is unlikely that they will overcome the limitations of Hebbian learning compared to supervised learning.

One of the key differences between the brain and networks described in this paper and deep nets is that the brain is highly recurrent. Feed forward nets, even the simple systems described in this paper, are capable of learning sophisticated categorisers. On the other hand recurrence is not well understood, and modelling it is difficult.

One machine learning benefit that may arise from using brain based systems is handling time. The recurrence allows persistent firing that can act as short term memory allowing the system to learn time based tasks more effectively.

Modern machine learning has been inspired by the brain, and has used ideas, such as visual fields, to improve performance. However, it has used mechanisms, such as supervised learning, that are not used in the brain, though there are attempts to integrate backpropagation with biology [12]. These non-biological mechanisms are mathematically solid and have led to improved systems, but still have not led to Turing test passing AI [19]. Perhaps this is because learning one task, such as face recognition, can take advantage of large marked up data to set millions of parameters, connection weights, in a deep net. Passing the Turing test, on the other hand, requires a system that can solve a practically infinite number of problems. Furthering understanding of how the brain does these tasks by modelling the brain at the neural level, may be the best way to develop a Turing test passing AI.

References

1. Bache, K., Lichman, M.: UCI machine learning repository (2013), http://archive.ics.uci.edu/ml
2. Belavkin, R., Huyck, C.: Conflict resolution and learning probability matching in a neural cell-assembly architecture. Cognitive Systems Research **12**, 93–101 (2010)
3. Bi, G., Poo, M.: Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. Journal of neuroscience **18**(24), 10464–10472 (1998)
4. Bienenstock, E., Cooper, L., Munro, P.: Theory for the development of neuron selectivity: orientation specificity and binocular interaction in the visual cortex. Jouranl of Neuroscience **2:1**, 32–48 (1982)
5. Brette, R., Gerstner, W.: Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. J. Neurophysiol. **94**, 3637–3642 (2005)

6. Davison, A., Yger, P., Kremkow, J., Perrinet, L., Muller, E.: PyNN: towards a universal neural simulator API in python. BMC neuroscience **8**(S2), P2 (2007)
7. Gewaltig, M., Diesmann, M.: NEST (NEural Simulation Tool). Scholarpedia **2**(4), 1430 (2007)
8. Hebb, D.: The organization of behavior: A neuropsychological theory. New York: Wiley (1949)
9. Hinton, G., Osindero, S., Teh, Y.: A fast learning algorithm for deep belief nets. Neural Computation **18:7**, 1527–1554 (2006)
10. Hodgkin, A., Huxley, A.: A quantitative description of membrane current and its application to conduction and excitation in nerve. Journal of Physiology **117**, 500–544 (1952)
11. Huyck, C., Mitchell, I.: Post and pre-compensatory Hebbian learning for categorisation. Computational Neurodynamics **8:4**, 299–311 (2014)
12. Lillicrap, T., Santoro, A., Marris, L., Akerman, C., Hinton, G.: Backpropagation and the brain. Nature Reviews Neuroscience **21:4**, 1–12 (2020)
13. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A., Veness, J., Bellemare, M., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature **518**, 529–33 (2015)
14. Rumelhart, D., McClelland, J.: Parallel Distributed Processing. MIT Press (1986)
15. Sejnowski, T., Koch, C., Churchland, P.: Computataional neuroscience. Science **241:4871**, 1299–1306 (1988)
16. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., Hassabis, D.: Mastering the game of go without human knowledge. Nature **550**, 354–59 (2017)
17. Song, S., Miller, K., Abbott, L.: Competitive hebbian learning through spike-timing-dependent synaptic plasticity. Nature neuroscience **3:9**, 919–926 (2000)
18. Sun, Y., Liang, D., Wang, X., Tang, X.: Deepid3: Face recognition with very deep neural networks. CoRR **abs/1502.00873** (2015), http://arxiv.org/abs/1502.00873
19. Turing, A.: Computing machinery & intelligence. Mind **59**, 433–460 (1950)
20. Wade, J., McDaid, L., Santos, J., Sayers, H.: Swat: a spiking neural network training algorithm for classification problems. IEEE Transactions on Neural Networks **21:11**, 1817–1830 (2010)