

Designer Concept Based Classification for Explainable Artificial Intelligence

Andrew Swinn¹ and Marek Grzes¹

¹School of Computing, University of Kent, UK.

Contributing authors: a.swinn@kent.ac.uk; m.grzes@kent.ac.uk;

We propose a novel approach to explainable artificial intelligence (XAI) that uses concept channels to build neuro-symbolic classifiers. The concept channels use neural techniques to extract interpretable representations from raw data. Symbolic techniques are then used to reason over them. By doing this, we leverage the strengths of the two worlds, namely the ability of neural learning models to discover hidden patterns and to tolerate uncertainty in data, and the innate explainability and interpretability of symbolic programs.

This work focuses on the classification task and introduces a new concept-based neural architecture. Using background knowledge, neural channels are each trained to detect the presence (or absence) of a single concept. Together, they yield a concept probability vector which, when transformed by a class concept map, gives a class probability vector from which predictions can be made. Re-application of the trained neural channels to the training data offers additional background knowledge in the form of class concept mappings, as well as a set of concept probabilities over which Naïve Bayes reasoning is performed.

The transformation between concepts and classes bridges the gap between black box neural representations and the reasoning required to produce a classification. Furthermore, because each concept channel only detects one concept, it can produce its own relevance map showing where its concept was found.

1 Introduction

Artificial intelligence (AI) is transforming many aspects of our lives. AI is increasingly able to diagnose medical conditions as reliably as doctors [1], Large Language Models achieve remarkable performance on most natural language processing tasks, and a personalized e-learning experience can be delivered with minimal human input [2]. The success of AI is largely driven by neural networks, which are notoriously difficult

to interpret or explain. However, explanations are essential in high-stakes or safety-critical domains, such as healthcare and autonomous vehicles [3].

Explainable AI has become an important branch of both fundamental and applied research, with many approaches and techniques used [4–6]. Interpretability enables the plausibility with which a classification or prediction can be assessed. Without interpretability, the danger of misclassification is always present. Uninterpretable classifiers carry risk: making commercial or safety decisions learned from poor data or spurious examples raises the question of who is responsible if they get it wrong [5]? Without interpretability, it is difficult to trust a model, and if users do not trust a model or a prediction, they will not use it [7].

An approach to XAI is to design and build models that are explainable [8]. Such white or transparent box models including decision trees, linear models, and logic programs are considered explainable as long as their size is sufficiently small [4]. Deep-ProbLog [9] uses neural networks to learn the predicates for probabilistic Prolog. Another white box method with a symbolic component [10] embraced a rule-based interpretable classifier using Bayesian probabilistic analysis. These models provide transparency because it is possible to follow their decision-making processes.

In AI, human knowledge is often assumed or required in the form of theories [11] or heuristics [12]. Heuristics, designed using human understanding of a problem, can speed up convergence to a good or potentially optimal solution (e.g. admissible heuristics in A*), or well-vetted theories allow for answering causal questions from observational data [11]. In this paper, we show that a certain level of human input allows for accurate interpretability of machine learning models.

Deep neural networks are not symbolic, and there is no easy way to interpret their decision-making processes. The network itself is made up of a set of large multidimensional tensors that denote the weights associated with the edges that connect the nodes (artificial neurons). Accounting for individual weights, or explaining the function of any given node (or group of nodes), is the subject of ongoing research, e.g., using mechanistic interpretability [13].

Neural networks operate in high-dimensional vector spaces that cannot be immediately mapped to interpretable representations. It is difficult to know which features (pixels) of an image cause which activations in a network, and it is difficult to know which activations contribute to an overall classification. The problem is exacerbated by the existence of poly-semantic neurons which carry more than one type of knowledge, and because individual (groups of) features may be common across many classes. To illustrate, Figure 1 gives 10 examples of digits in the MNIST digit test set that all have a common feature: the value of pixel $(14, 14) = 252$. It is the relationship between pixel values as much as the individual pixel values themselves that carry meaning; the

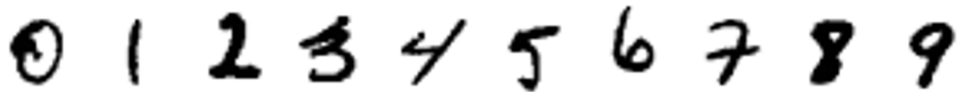


Fig. 1: Digits with a common pixel

information carried by an individual pixel depends on whether the pixel is part of a horizontal line or part of a vertical line.

This research offers an approach to classification that is designed to be, at least partly, explainable; it couples neural deep learning components with symbolic components to improve the quality of classification explanations. The symbolic components use human-understandable rules to reason over the outputs of the neural components. The neural components offer some degree of explainability by producing relevance maps and detecting concepts.

Our idea is to train a small neural model to map a highly dimensional image feature vector to a much lower dimensional concept vector and perform symbolic reasoning over this concept vector. Our approach differs from other concept-based approaches to explainability because we use human background knowledge to design the concepts, and then guide the neural components to detect them. This then leads to human-understandable reasoning. We also explore a sampling-based approach to see how human background knowledge performs against random background knowledge.

2 Related Work

The explainability problem arises because of the sub-symbolic representation of knowledge within a neural network. Artificial neural networks are trained by adjusting the weights between large numbers of neurons (nodes) in a network. Even small networks often contain large numbers of neurons; the comparatively small LeNet5 network [14] classified the MNIST digit dataset with a network composed of 6×10^5 nodes. Large language models have 10's or 100's of billions of nodes. The internal representation of an input vector is made up of the node activations across the network. With so many nodes, the mapping between the input vector, the internal representation, and the output classification is unintelligible to humans.

Several post hoc approaches have been proposed to explain how neural networks and other black box models make a certain decision. LIME [7] produces local linear explanations by computing a simple model that approximates the given model near a data point. SHAP [15] gives feature rankings based on their marginal contributions to a classification. There are also techniques for deep multi-layered neural networks that produce saliency maps highlighting which part of an image contributes most to a classification [16, 17]. These techniques can be applied to existing pre-trained neural models but do not guide the models themselves to be inherently explainable.

A second set of approaches to explainability has been to integrate neural techniques with symbolic techniques to create an architecture which exploits the best characteristics of both types of classifiers. Symbolic techniques use explicit representations that are understandable by a human [18]. One symbolic technique is naïve Bayes reasoning which uses evidence, prior probabilities and likelihoods. A second symbolic approach is linear regression using data to find linear relationships. There are different neuro-symbolic architectures, including [Symbolic $\rightarrow$ Neural $\rightarrow$ Symbolic] in which neural processing is embedded within a symbolic system, and [Neural[Symbolic]] in which symbolic reasoning is embedded within the neural system. Two approaches to neuro-symbolic integration emerge [19]: learning for reasoning in which a neural component

preprocesses input data to make it more suitable for symbolic reasoning (e.g., neural processing could reduce dimensionality before a symbolic tree search), and reasoning for learning in which symbolic preprocessing is performed to aid the neural learning. Typically, neuro-symbolic integration has four components [20], each of which is present in our approach:

1. translation of symbolic (background) knowledge into the network,
2. learning of additional knowledge from examples (and generalisation) by the network,
3. executing the network (i.e. reasoning),
4. symbolic knowledge extraction from the network.

Complimenting these approaches to explainability are a set of concept-based methods. Concepts are emerging as human understandable characteristics of highly dimensional feature vectors in which the individual features have no intrinsic meaning (illustrated by Figure 1). The semantic meaning of such a data example is carried by human-understandable concepts, not by the machine-understandable features. A growing number of concept-based approaches are being explored [21]. They can be grouped together using three different types of concept-based explanations:

1. **class-concept** relations in which concepts (e.g. ‘loop’) are mapped to a subset of classes (e.g. a loop commonly found in the 0, 6, 8 and 9 classes of the MNIST Digits dataset) [22–25];
2. **node-concept** associations where individual network nodes are chosen to represent concepts (training them using concept loss and backpropagation) [26–29];
3. **concept visualizations** - using techniques like saliency maps applied to individual concepts [30–32].

Concept-based explanations can be defined as functions g mapping machine explanations (vectors in a machine explanation space E_m) to human understandable explanations (vectors in the human explanation space E_h) [22]:

$$g : E_m \rightarrow E_h. \quad (1)$$

This definition and the fact that machine-based explanations and human-based explanations can be quite different reveals the difficulty in producing successful XAI. While machines can be very successful in imitating and even outperforming humans, they can do so based on evidence and reasoning that is completely different from that used by humans. By using such reasoning, they are vulnerable to confounding factors [11], as well as adversarial attacks (undetectable input perturbations designed to change a classification). Moreover, explanations based on such artificial reasoning can also be unnecessarily complex, making them less persuasive to a critical human audience.

Another complexity hidden by Equation 1 is that, for a given model, there can be many different explanations suited to different humans with different ways of understanding the world or to different scientific theories [11].

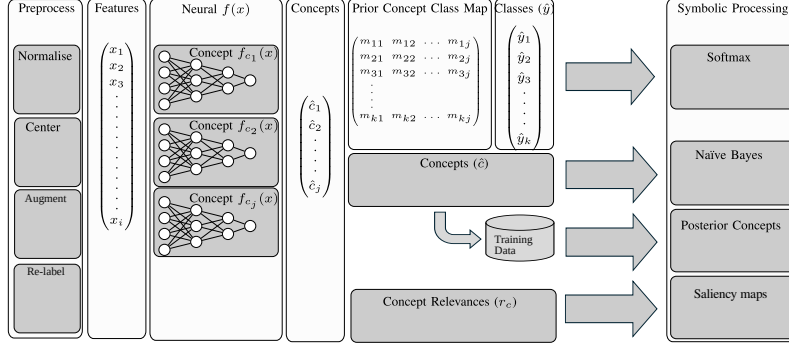


Fig. 2: Designer Concepts Architecture

Concepts are similar to but different from latent features. In order to enhance explainability, concepts must be visible (and ideally obvious) to humans. Constraining representations to express a single well-defined concept in a single network node is difficult; for example, concept methods are known to experience concept leakage [33] or a concept whitening module may be needed [34] to attempt to disentangle concept representations. Our use of concept channels is intended to avoid the need for whitening and prevent concept leakage altogether.

Finally, the original motivation for this work was the intuition that neural networks, because of their dependence on random initialisations and huge amounts of training data, are very difficult to test. The same network architecture trained on the same data will produce a different set of connecting weights. Although work to develop mechanistic interpretability of neural networks is ongoing, the role of individual nodes in a neural network is subject to luck [35]. Given these problems, it seemed that any approach that reduces the number of nodes in a neural part of the system ought to be easier to mechanistically interpret and so easier to fully test.

3 Method

The architecture used to implement the designer concepts approach is shown in Figure 2. Input data is preprocessed (normalised, centred, augmented etc.) to create a feature vector ($x \in \mathbb{R}^n$) suitable for neural processing. The feature vector is then propagated through the parallel feedforward neural concept channels $f(x) = (f_{c_1}(x), f_{c_2}(x) \dots f_{c_m}(x))$ to give a concept probability vector ($\hat{c} \in \mathbb{R}^m$). The concept probability vector is then symbolically translated to an unnormalised class prevalence vector $\hat{y} \in \mathbb{R}^k$, and then to a class probability vector ($\hat{p}(y) \in \mathbb{R}^k$) using the softmax. The model provides three outputs:

1. Class prediction ($\hat{p}(y)$): the class probability vector lets us predict the most likely class.
2. Concept prediction ($\hat{c}$): the concept probability vector is provided directly from the neural channels. Exposing the concept vector allows us to see which concepts have been detected (or missed).

Digit	Cross	T Junction	Loop	LineUp	LineDown	TopLeft	BottomLeft	TopRight	BottomRight
0			x						
1				x	x				
2						x			x
3						x	x		
4		x		x	x				
5							x	x	
6		x	x	x					
7					x	x			
8	x		x						
9		x	x		x				

Table 1: Prior background knowledge

3. Concept relevances: concept relevances produce saliency maps similar to layer-wise or concept relevance propagation [17, 23]. Our relevances differ from those techniques in that they are produced directly from the knowledge representations within the neural channels.

3.1 Neural Training

To train the neural channels, a human-generated prior class concept map ($\mathcal{M} \in \mathbb{R}^{k \times m}$), based on background knowledge (Table 1) is used to map labelled training data to concept vectors, $y \rightarrow c$ (e.g. all 0's, 6's, 8's and 9's have a loop). After training, it is used to translate predicted concept vectors back to class probabilities ($\hat{c} \rightarrow \hat{y} \rightarrow \hat{p}(y)$).

Each of the neural concept channels implements a function $f_{c_n}(x)$ mapping the normalised input vector to a single concept probability. The channels have been constrained to be entirely independent of each other to prevent concept leakage [33] and were kept as simple as possible. For more complex classification tasks, the channels could be built upon a shared pre-trained feature detector or have customised neural architectures based on the nature of their target concept.

Each channel implements a convolutional layer $[g_c * x]$ followed by a pooling layer ($\mathcal{P}_{max}$) and fully connected layer w_c followed by a sigmoid activation. The overall concept activation is defined by Equation 2.

$$f_{c_n}(x) = \sigma \left(\sum_{i,j} (w_{c_n(i,j)} \cdot \mathcal{P}_{max}([g_{c_n} * x])_{(i,j)}) \right)^1 \quad (2)$$

Concept relevances $r_{c_n}(x)$ are produced by taking the product of the pooling layer $\mathcal{P}_{Max}([g_c * x])$ and the weights matrix (w_c), and retaining the positional indices (i, j) .

¹ i, j represent the position of each node in the pooled output

The positional indices (i, j) corresponding to the nodes in the pooled layer are scaled up to match the pixel coordinates of the input image. In this way, the concept relevances, $r_{c_n}(x)_{(i,j)}$, show exactly which regions of an image contribute to a concept identification:

$$r_{c_n}(x)_{(i,j)} = w_{c_n(i,j)} \cdot \mathcal{P}_{max}([g_{c_n} * x])_{(i,j)}. \quad (3)$$

The concept activation Equation 2 can be restated in terms of concept relevance:

$$f_{c_n}(x) = \sigma\left(\sum_{i,j} r_{c_n}(x)_{(i,j)}\right). \quad (4)$$

Each of the three neural outputs implements a differentiable function, so each of the three outputs can have a loss function associated with it.

1. Class predictions: a categorical loss function is used to direct the network to produce correct class predictions over k classes:

$$\mathcal{L}_{Class}(x) = -\sum_{c=1}^k y_{o,c} \log(p_{o,c}).$$

2. Concept predictions: a regression loss² can be used to direct each of the m concept channels predict the presence of concepts directly (using the background knowledge contained in the concept feature map):

$$\mathcal{L}_{Concept}(x) = \sum_{i=1}^m (\hat{c}_i - c_i)^2.$$

3. Concept relevances: a focus loss can be used to encourage the channel classifiers to find hotspots:

$$\mathcal{L}_{Focus}(x) = \frac{\mathcal{P}_{avgtotal}(r_c^+(x))}{\mathcal{P}_{maxtotal}(\mathcal{P}_{avg_{n \times m}} r_c^+(x))}.$$

The focus loss function $\mathcal{L}_{Focus}(x)$ is evaluated on the 2 dimensional relevance $r_c(x)$. It operates on the positive contributions in the relevance (allowing negative contributions to remain diffuse), and by guiding the network to minimise the ratio of average temperature to maximum temperature, it promotes relevance with a clear focus and discourages relevance with a diffuse focus. The shape of the average pooling kernel ($n \times m$) is used to determine the sharpness of the focus. Figure 3 gives examples of unfocused and focused relevance maps. The focused maps show far less ‘noise’, particularly on the 0, 6, 8 and 9 examples.

The modular nature of the architecture, the embedding of human background knowledge, and the ability to choose where to backpropagate losses gives a good deal of control over the neural representations learned by the concept channels. It lets us make those representations explainable.

²A binary cross entropy loss could also have been used.

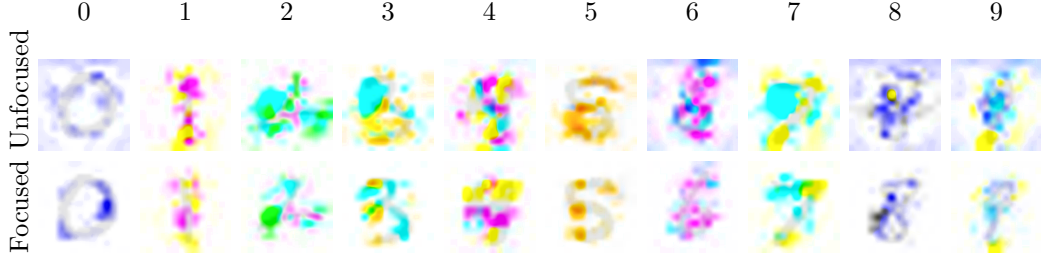


Fig. 3: Unfocused and focused relevances

3.2 Symbolic Processing

The output of our neural concept predictors is processed by three distinct symbolic approaches, mapping concepts to the final classes of the target classification task.

The first approach uses the prior class concept map $\mathcal{M}$ (Table 1) to translate from the concept probability vector to a class prevalence vector $\hat{y}$ ($\hat{c} \rightarrow \hat{y}$) followed by the softmax function (Equation 5) to convert the logit $\hat{y}$ vector into a class probability vector $\hat{p}(y)$ used to select the most probable class.

$$\hat{p}(y_i) = \frac{e^{\hat{y}_i}}{\sum_i e^{\hat{y}_i}} \quad (5)$$

The second, naïve Bayes, approach uses Equation 6 for Bernoulli-distributed predictors to find the most probable class. The prior class probabilities, $p(y_i)$, and the conditional concept likelihoods, $p(c_j|y_i)$, are found by re-applying the trained concept channels onto the training data. The concept evidence, $p(c_j)$, is found using the trained concept channels on the test data, where each concept channel gives the probability that each concept is present in each test data item. The probability of a concept being absent is represented as $p(\bar{c}_j)$.

$$p(y_i|c) \propto p(y_i) \prod_j \left[p(c_j|y_i)^{p(c_j)} + (1 - p(c_j|y_i))^{p(\bar{c}_j)} \right] \quad (6)$$

A human understandable basis for the classification can be found by inspecting the concept vector ($\hat{c}$).

The final ‘Posterior Concepts’ approach develops and extends the background knowledge, analysing the relationship between the learned concepts and the class attribute of the training data. The trained concept channels are re-applied to the training data to get a set of new concept vectors ($\hat{c}$). Each of these new concept vectors will have been detected in a subset of the training images and is mapped to the most common label of the images in that subset. These mappings to the most common class constitute our third classification method. This technique allows for classifications to be explained by showing the similar items in the training data. It also provides a degree of confidence in the classification based on the strength of the majority. It could also allow for unrepresentative training data to be removed from the training data set.

Filters	Parameters	Accuracies (%) by Loss Type			
		Explain	Prediction	Concept	Both
1	2205	None	91.5	93.7	92.4
2	4410		94.8	96.3	95.7
4	8820		96.0	97.0	97.2
8	17640		97.0	97.7	97.5
16	35280		97.0	97.4	97.3
1	2205	Focus	91.0	93.8	93.1
2	4410		94.9	95.9	95.6
4	8820		96.2	97.2	96.6
8	17640		96.4	97.3	97.2
16	35280		96.5	97.6	96.6

Table 2: Testing Accuracies

4 Implementation and Experiments

4.1 Data and Background Knowledge

The MNIST Digits dataset [36] was chosen for the experiments incorporating background knowledge into a network. It was partly chosen because it has been well studied; comparing results against other studies is straightforward [14, 37]. It was also chosen because of the simplicity of deriving and expressing the prior background knowledge (1’s contain a line pointing up towards the top of the digit, and a line pointing towards the bottom of the digit. 6’s contain a loop and a line pointing up while 9’s contain a loop and a line pointing down). The full set of background knowledge is given in Table 1. The prior class concept map $\mathcal{M}$ translates the background knowledge into a matrix. This matrix implements the symbolic translation of probabilistic concept vectors $\hat{c}$ into probabilistic class vectors $\hat{y}$:

$$\hat{y} = \mathcal{M} \cdot \hat{c}. \quad (7)$$

4.2 Training the Neural Component

The neural components were created with different numbers of 7×7 convolutional filters followed by a 2×2 pooling layer and trained with losses applied in combinations to the predictions, concepts and relevance outputs over 10 epochs. Classification accuracies were measured by comparing labels (y_i) and the result of the softmax function (Equation 5) applied to the class vector ($\hat{y}$) output of the architecture. Table 2 shows the accuracies on test data.

The table shows that very small models (1 or 2 convolutional filters) perform relatively poorly, but with 4 or more filters the models classify consistently. It shows that models trained just using a concept loss perform marginally better (approx. 1%) than those models trained with just a class prediction loss. This can be understood because concept losses are applied directly to the concept channels rather than indirectly (where the losses must be first backpropagated through the symbolic prior class concept map).

Using both prediction and concept losses seems to result in an accuracy somewhere between the two. There is very little variance between the accuracies of the focused and unfocused relevance maps, and across the different prediction types suggesting little trade-off between explainability and accuracy.

4.3 Searching for Background Knowledge

A second experiment sought to remove the dependence on prior background knowledge by finding an alternative prior class concept map to that specified in Table 1. Two approaches to the search were considered: a tree search in which concept maps were expanded by one concept at a time, and a random approach, where class concept maps were randomly guessed. The idea of this search was to establish the extent to which the overall performance of the architecture depends on good background knowledge. The accuracy of the classifiers with randomly guessed prior concept maps was lower than the classifier with the handcrafted prior concept map but not by much. Averaged over 2624 separate random experiments, the classification accuracy with search methods was $94.9 \pm 0.8\%$. The very low variance of these classifiers militates against using a tree search because it is lower than the variance caused by the random initialisation of neural networks. While the tree search would follow the best path, the result would be statistically no different from random sampling.

To maintain the quality of the random prior concept maps, they were sampled with different numbers of concepts and different levels of sparsity (the probability of a concept mapping to a class). There was also an admissibility constraint applied to the prior concept map - no two classes could be mapped to the same set of concepts.

There was a weak correlation between accuracy and the number of concepts (4 concepts gave a mean accuracy of 93.5%, while 8 concepts gave an accuracy of 95.2%). The sparsity parameter made no difference. This experiment indicates that many concept maps are feasible on this data, and domain knowledge is necessary to determine the right (most useful) one, which agrees with the challenges of scientific enquiry, in general [11].

4.4 Comparative Accuracies

Overall, the accuracies compare well to many techniques but lag behind the top performers. LeNet5 [14] achieved an accuracy of 99% on the MNIST dataset. In their review of MNIST classifiers [37] compare 5 neural architectures that achieve testing accuracies ranging from 98.4% to 99.6% - between 1% and 2% higher than our approach. They report 13 non-neural approaches which all failed to reach the accuracy of our approach. Comparing the accuracies with and without the custom focus loss suggests that applying a focus loss has no overall effect on predictive accuracies. Some classifiers perform marginally better, while others perform marginally worse than their unfocused counterparts.

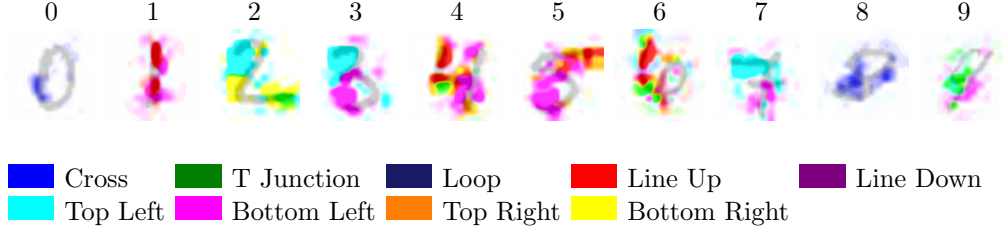


Fig. 4: Saliency maps (r_c)

5 Explanation Results

5.1 Explanations using Background Knowledge

The architecture allows classifications to be made using the three symbolic approaches: using the softmax function applied to the class vector $\hat{y}$, using naïve Bayes prior and posterior probabilities, and using the posterior class concept map. The techniques give very similar accuracies ($97.5 \pm 0.3\%$). For comparison, a classifier using the LeNet5 convolutional architecture [14] was trained over 10 epochs using the same data. As well as providing a baseline for statistical comparisons of overall accuracy, the LeNet results allow us to analyse misclassifications of individual data items.

Analysis showed a substantial overlap between the misclassifications made by the different neuro-symbolic techniques (166 data examples are misclassified regardless of how the classification was made). It also lets us calculate the number of data items that were misclassified by at least 2 of the symbolic techniques (228). What was unexpected was that nearly half of the examples misclassified by the LeNet5 architecture (37 out of 96) were correctly classified by at least 1 of our classifiers.

Inspecting the input images, there was no obvious human detectable characteristic which led to a LeNet5 or neuro-symbolic misclassification.

5.1.1 Softmax Explanations

The softmax classifier uses the class probability vector $\hat{p}(y)$ (obtained from $\hat{y}$ using softmax) to choose the most likely class. This is the most successful technique but is the most difficult to explain because there is no direct link back to the training data. Instead, explaining these predictions relies on the relevance maps (r_c) which reveal which part of an image each of the concept channels is detecting (Figure 4). From the images, it is clear which part of an image is contributing to the identification of a concept. The ‘2’ has a yellow area corresponding to the bottom-right concept and a cyan area corresponding to the top-left. This concept is working as intended. The red Line Up concept is clearly present in the ‘1’, the ‘4’ and the ‘6’. The ‘4’, ‘6’ and ‘9’ all have areas of green corresponding to a T junction in roughly the right areas.

However, it is difficult to see where the loop was identified in the ‘6’ and ‘9’, and there is no sign of the cross in the ‘8’. It might be because a faint dark blue looks like a clear navy blue.

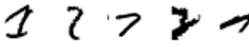
Prediction	Training Images	Prediction	Training Images
0		5	
1		6	
2		7	
3		8	
4		9	

Fig. 5: Posterior Class Concept Map

5.1.2 Naïve Bayes Classification Explanations

The Naïve Bayes classification uses Bayes rule (Equation 6) to find the most probable class. Inspecting the probabilities showed values of $p(c_j|y_i) \leq 0.02$ (where the background knowledge indicates the lack of a concept) and $p(c_j|y_i) \geq 0.95$ (when a concept should be present); the neural classifiers are very good at detecting what they are told to detect (Table 1). The prior probabilities of the digits $p(y_i) \approx 0.10$ because the training data is balanced across classes.

5.1.3 Posterior Concept Map Explanations

The Posterior Concept Map is derived by re-applying the trained neural architecture onto the training data to give a concept probability vector $\hat{c}$ to each training example. This allows training data items which do not fall into the definitions described by the prior background knowledge to be used, adding to the background knowledge. In the training data, there were 105 different concept probability vectors (95 new concept probability vectors not found in the prior concept map). In the Posterior Concept Map, for each concept probability vector, the predicted label is the most common label of those training examples sharing the concept probability vector $\hat{c}$. Figure 5 gives examples of images with concept configurations not found in the prior concept map. Each row contains digits sharing a concept configuration. The '0' row shows images with the loop and an additional top-left concept. The '7' row shows images with the line down and top left concept, and an additional loop concept.

The most striking thing about the images in table 5 is the number of those images that are clearly ambiguous. In the '0' row, there are two mirrored 6's. In the 4's row half of the images could easily be a 4 or a 9.

5.2 Sensitivity of Concept Activation Vectors

Finally, the concepts used in the initial prior class concept map (Table 1) were used to see if they could be found in the LeNet5 network. Using the fully connected hidden layer with 128 nodes and with the presence or absence of concepts denoted by the prior class concept map, a simple neural network was trained to search for Concept Activation Vectors (CAVs) [22]. Interestingly, our prior concepts were indeed found in

the LeNet5 architecture. However, when the experiment was repeated with all possible concept maps (there are 2^{10} possible concept maps - each corresponding to a different subset of the 10 digits) most of them could be found as well (demonstrating the high recall and low precision of CAVs).

6 Discussion

This novel approach to building explainable classifiers presents the opportunity to build neural classifiers that have explainable reasoning, and which have a comparatively low number of neurons organised into distinct channels. A channel comprising a few thousand, or even a few hundred neurons ought to be much more tractable for other explainability techniques (e.g. automated circuit discovery [13]). It offers a direct view back onto the training data through the posterior class concept explanations - which would provide concrete grounds for a classification and would allow unhelpful, or erroneous training data to be identified and removed. While the prior class concept map is implemented as a fixed layer in a neural layer allowing errors to be back-propagated through to the concept channels, it could be updated (i.e. learned) through back-propagation, or updated with posterior concept mappings during the training process.

More generally, the use of background knowledge to structure classification subsystems could be applied to any set of machine learning techniques. However, in domains where background knowledge is unavailable or in a form difficult to represent, this technique may replace one unexplainable classifier with a different unexplainable classifier.

Finally, the real ambiguity of many of the data items revealed by this approach suggests that the very high claims of accuracy made by some classifiers could be misleading, especially if they exceed expert human classifiers on such a simple task.

References

- [1] A. Baker, Y. Perov, Middleton Katherine, J. Baxter, D. Mullarkey, D. Sangar, M. Butt, DoRosario Arnold, and S. Johri, “A Comparison of Artificial Intelligence and Human Doctors for the Purpose of Triage and Diagnosis,” *Frontiers in Artificial Intelligence*, vol. 3, 2020.
- [2] J. Jin and M. Kim, “GPT-Empowered Personalized eLearning System for Programming Languages,” *Applied Sciences*, vol. 13, no. 23, p. 12773, 2023.
- [3] P. Kouvaros, T. Kyono, F. Leofante, A. Lomuscio, D. Margineantu, D. Osipychiev, and Y. Zheng, “Formal Analysis of Neural Network-Based Systems in the Aircraft Domain,” in *Formal Methods* (M. Huisman, Păsăreanu Corina, and N. Zhan, eds.), (Cham), pp. 730–740, Springer International Publishing, 2021.
- [4] A. A. Freitas, “Comprehensible classification models: a position paper,” *SIGKDD Explor. Newsl.*, vol. 15, pp. 1–10, 3 2014.
- [5] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, Giannotti Fosca, and D. Pedreschi, “A Survey of Methods for Explaining Black Box Models,” *ACM Comput. Surv.*, vol. 51, 8 2018.

- [6] Y. Zhang, P. Tino, A. Leonardis, and K. Tang, “A Survey on Neural Network Interpretability,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 5, pp. 726–742, 10 2021.
- [7] M. T. Ribeiro, S. Singh, and C. Guestrin, “Why Should I Trust You?: Explaining the Predictions of Any Classifier,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’16, (New York, NY, USA), pp. 1135–1144, Association for Computing Machinery, 2016.
- [8] C. Rudin, “Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead,” *Nature Machine Intelligence*, vol. 1, pp. 206–215, 8 2019.
- [9] R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester, and L. De Raedt, “DeepProbLog: Neural Probabilistic Logic Programming,” in *Advances in Neural Information Processing Systems* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), vol. 31, Curran Associates, Inc., 2018.
- [10] B. Letham, C. Rudin, T. H. McCormick, and D. Madigan, “Interpretable classifiers using rules and Bayesian analysis: Building a better stroke prediction model,” *The Annals of Applied Statistics*, vol. 9, 9 2015.
- [11] J. Pearl, *Causality*. Cambridge University Press, 2 ed., 2009.
- [12] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Pearson, 3rd ed., 2014.
- [13] A. Conmy, A. Mavor-Parker, L. Aengus, S. Heimersheim, and Adrià Garriga-Alonso, “Towards Automated Circuit Discovery for Mechanistic Interpretability,” 2023.
- [14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [15] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” in *Proceedings of the 31st International Conference Neural Information Processing Systems*, NIPS’17, (Red Hook, NY, USA), pp. 4768–4777, Curran Associates Inc., 2017.
- [16] K. Simonyan, A. Vedaldi, and A. Zisserman, “Deep inside convolutional networks: Visualising image classification models and saliency maps,” *arXiv preprint arXiv:1312.6034*, 2013.
- [17] Bach Sebastian, A. Binder, Montavon Grégoire, Klauschen Frederick, Müller Klaus-Robert, and Samek Wojciech, “On Pixel-Wise Explanations for Non Linear Classifier Decisions by Layer-Wise Relevance Propagation,” *PLOS ONE*, vol. 10, pp. 1–46, 8 2015.
- [18] M. K. Sarker, L. Zhou, A. Hitzler, and Eberhartand Pascal, “Neuro-Symbolic Artificial Intelligence: Current Trends,” 2021.
- [19] D. Yu, B. Yang, D. Liu, H. Wang, and S. Pan, “A survey on neural-symbolic learning systems,” 9 2023.
- [20] T. R. Besold, A. d’Avila Garcez, S. Bader, H. Bowman, P. Domingos, P. Hitzler, K.-U. Kuehnberger, L. C. Lamb, D. Lowd, P. M. V. Lima, L. de Penning,

- G. Pinkas, H. Poon, and G. Zaverucha, “Neural-Symbolic Learning and Reasoning: A Survey and Interpretation,” 2017.
- [21] E. Poeta, G. Ciravegna, E. Pastor, T. Cerquitelli, and E. Baralis, “Concept-based Explainable Artificial Intelligence: A Survey,” 12 2023.
 - [22] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas, and R. Sayres, “Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV),” tech. rep., Proceedings of the 35th International Conference on Machine Learning, 2018.
 - [23] R. Achtabat, M. Dreyer, I. Eisenbraun, S. Bosse, T. Wiegand, W. Samek, and S. Lapuschkin, “From attribution maps to human-understandable explanations through Concept Relevance Propagation,” *Nature Machine Intelligence*, vol. 5, pp. 1006–1019, 9 2023.
 - [24] J. Crabbé and M. van der Schaar, “Concept Activation Regions: A Generalized Framework For Concept-Based Explanations,” *Proceedings of the 36th International Conference on Neural Information Processing Systems*, 9 2022.
 - [25] Y. Goyal, A. Feder, U. Shalit, and B. Kim, “Explaining Classifiers with Causal Concept Effect (CaCE),” 7 2019.
 - [26] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba, “Network Dissection: Quantifying Interpretability of Deep Visual Representations,” *Conference on Computer Vision and Pattern Recognition*, pp. 6541–6549, 4 2017.
 - [27] R. Fong and A. Vedaldi, “Net2Vec: Quantifying and Explaining how Concepts are Encoded by Filters in Deep Neural Networks,” 1 2018.
 - [28] B. Zhou, D. Bau, A. Oliva, and A. Torralba, “Interpreting Deep Visual Representations via Network Dissection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, pp. 2131–2145, 9 2019.
 - [29] P. W. Koh, T. Nguyen, Y. S. Tang, S. Mussmann, E. Pierson, B. Kim, and P. Liang, “Concept Bottleneck Models,” 2020.
 - [30] A. Ghorbani, J. Wexler Google Brain, J. Zou, and B. Kim Google Brain, “Towards Automatic Concept-based Explanations,” tech. rep., 2019.
 - [31] C.-K. Yeh, B. Kim, S. Arık, C.-L. Li, T. Pfister, and P. Ravikumar, “On Completeness-aware Concept-Based Explanations in Deep Neural Networks,” tech. rep., 2020.
 - [32] B. Zhou, Y. Sun, D. Bau, and A. Torralba, “Interpretable Basis Decomposition for Visual Explanation,” tech. rep., 2018.
 - [33] A. Mahinpei, J. Clark, I. Lage, F. Doshi-Velez, and W. Pan, “Promises and pitfalls of black-box concept learning models,” *arXiv preprint arXiv:2106.13314*, 2021.
 - [34] Z. Chen, Y. Bei, and C. Rudin, “Concept whitening for interpretable image recognition,” *Nature Machine Intelligence*, vol. 2, pp. 772–782, 12 2020.
 - [35] J. Frankle and M. Carbin, “The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks,” 2019.
 - [36] Y. LeCun, C. Cortes, and C. Burges, “The MNIST database of handwritten digits,” 1998.
 - [37] S. S. Kadam, A. C. Adamuthe, and A. Patil, “CNN Model for Image Classification on MNIST and Fashion-MNIST Dataset,” *Journal of scientific research*, 2020.