CONVOLUTION NEURAL NETWORK POWERED PROBABILISTIC CIRCUIT FOR IMAGE PROCESSING

Weronika Kopytko (s244446) & Miłosz Holeksa (s246472)
Project supervisor: Niccolò Ciolli

ABSTRACT

This study investigates a hybrid architecture combining Convolutional Neural Networks (CNNs) with Probabilistic Circuits (PCs) to enhance performance on image classification tasks. CNNs are effective at extracting spatial features from images, capturing patterns from edges to complex textures through hierarchical filters. PCs, in contrast, are probabilistic models capable of exact inference and robust handling of uncertainty and missing data. The architecture leverages a CNN for feature extraction, transforming image data into structured representations for PCs, which model probabilistic dependencies to classify handwritten digits from the MNIST dataset. Results show improved classification accuracy and robustness to incomplete data compared to baseline PC model, demonstrating the potential of combining CNNs and PCs for tasks requiring interpretability and uncertainty handling.

***Index Terms*—** Probabilistic circuit, CNN, SPN, image classification

1. INTRODUCTION

Probabilistic models are at the core of modern machine learning and artificial intelligence, providing a guiding framework for decision making in the presence of uncertainty. They allow for robust reasoning by modeling data as originating from an underlying probability distribution. Sum-product networks (SPNs) and their generalization into Probabilistic Circuits (PCs) have emerged as powerful tools in this paradigm, enabling efficient probabilistic inference and learning. PCs can be viewed as both probabilistic graphical models and deep networks. They can efficiently execute various probabilistic inferences in a single forward and backward pass, with the ability to naturally marginalize missing data [1, 2].

Although PCs excel in interpretability and uncertainty modeling, their application to structured data such as images is challenging because of their fixed structure. Convolutional Neural Networks (CNNs), in contrast, are widely regarded as the state-of-the-art architecture for image processing tasks [3]. Their hierarchical design and localized filters enable the efficient extraction of spatial features, capturing both low-level patterns (e.g., edges) and high-level representations (e.g., textures). However, CNNs lack the ability to explicitly model uncertainty, which can limit their interpretability in critical domains.

This paper explores the integration of CNNs with PCs to leverage their complementary strengths. The proposed hybrid architecture employs CNNs to preprocess image data, extracting structured feature representations that serve as input to a PC for classification. By combining CNNs' feature extraction capabilities with PCs' probabilistic reasoning and exact inference, this approach aims to improve classification accuracy while enhancing robustness to incomplete data and interpretability. Specifically, we evaluate the architecture on the MNIST dataset, focusing on handwritten digit classification.

By demonstrating the model's ability to handle uncertainty and incomplete data while achieving high classification accuracy in the MNIST data set, this research establishes a robust framework for image classification tasks that require interpretability and resilience. Furthermore, it presents a novel approach for applying hybrid architectures to complex domains where probabilistic inference and hierarchical feature learning are critical.

2. PROBABILISTIC CIRCUITS (PCS)

Probabilistic Circuits (PCs) are structured probabilistic models designed to perform efficient and exact inference. They define joint probability distributions over a set of random variables through computational graphs. These graphs consist of three main components: *leaf nodes*, *product nodes*, and *sum nodes*. Leaf nodes model base distributions (e.g., Bernoulli, Gaussian or Categorical), product nodes represent factorizations of independent variables, and sum nodes encode weighted mixtures of distributions. The hierarchical structure of PCs allows for efficient and exact inference, such as marginalization and conditional probabilities, in time linear to the size of the circuit. The architecture ensures that PCs are both scalable and theoretically grounded, making them suitable for tasks involving uncertainty and missing data [2].

2.1. Mathematical Foundations of PCs

1. Leaf Nodes: Leaf nodes represent univariate probability distributions. For example, a Bernoulli leaf node which was used in our PC model, for a binary variable X_i computes:

$$P(X_i) = p_i^{X_i} \cdot (1 - p_i)^{1-X_i}$$

where p_i is the probability of $X_i = 1$.

A categorical leaf node for a class variable Y with K categories computes:

$$P(Y = k) = \pi_k$$

where π_k is the learned probability of class k.

2. Product Nodes: Product nodes assume independence between their child nodes. For child distributions P_1 and P_2, a product node computes:

$$P_{\text{prod}} = P_1 \cdot P_2$$

3. Sum Nodes: Sum nodes encode mixtures of child distributions. For child distributions P_1 and P_2, and weights w_1 and w_2, a sum node computes:

$$P_{\text{sum}} = w_1 \cdot P_1 + w_2 \cdot P_2$$

where $\sum w_i = 1$ and $w_i \geq 0$.

The circuit recursively combines these operations, propagating probabilities from the leaf nodes to the root node, which computes the final joint probability distribution.

3. MODELS

In order to investigate the performance of the hybrid architecture, we built three models:

1. CNN model
2. Mixed CNN-PC model
3. Baseline PC model

3.1. CNN model

The CNN architecture (see 1 and 2 in Figure 1) processes 28×28 grayscale images from the MNIST dataset through two convolutional layers, each followed by ReLU activation and max pooling. The first convolutional layer uses 3 filters of size 5×5, producing feature maps of size 24×24, which are downsampled to 12×12 using max pooling. The second convolutional layer applies 6 filters of size 5×5, resulting in feature maps of size 8×8, further reduced to 4×4 by max pooling. The final output is flattened into a 96-dimensional feature vector, which - after binarization - serves as input to the PC layers.

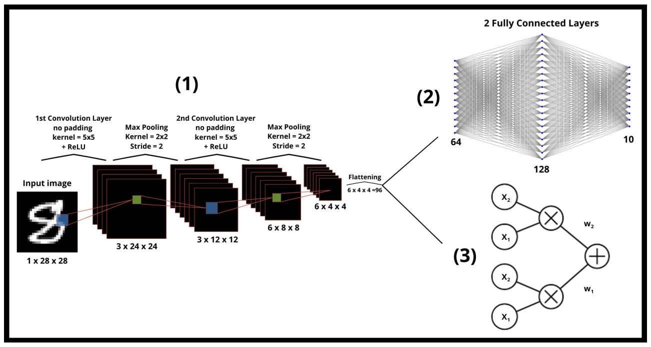

Fig. 1. Network structure of the CNN model (1 and 2) and Mixed CNN-PC model (1 and 3).

3.2. Mixed CNN-PC model

In the Mixed CNN-PC model (see 1 and 3 in Figure 1), the PC layers process CNN-extracted features to compute the joint likelihood of features and class labels. CNN in the Mixed CNN-PC model serves as a feature extractor, transforming raw image data into a structured representation suitable for probabilistic reasoning. The PC layers process binarized features extracted by the CNN using 82 channels c:

- **96 Bernoulli leaf nodes** for binary CNN features:

$$P(X_i \mid c) = p_i^{X_i} \cdot (1 - p_i)^{1-X_i}$$

- **1 categorical leaf node** for the class label k:

$$P(Y = k \mid c) = \pi_k$$

The product layer computes the joint likelihood for each channel c as:

$$P_c = \prod_{i=1}^{96} P(X_i \mid c) \cdot P(Y \mid c)$$

Finally, the weighted summation layer aggregates probabilities across channels:

$$z = a + \log \left(\sum_{c=1}^{82} \exp(P_c - a) \cdot w_c \right)$$

where $a = \max_c(P_c)$ ensures numerical stability. This step is fundamental to the probabilistic reasoning performed by the PC, producing normalized logits z that represent the probabilities for each class. The most probable class label is then inferred based on these probabilities.

3.3. Baseline PC model

The Baseline PC model has a similar architecture to the Mixed CNN-PC model with some differences. The input to the Baseline PC model consists of images from the MNIST dataset, sized 28x28. Each picture is flattened and passed as an input of size 785 (784 pixels + one label. Therefore, in order to keep the total number of parameters consistent with the Mixed CNN-PC model, the number of channels is equal to 10 for the Baseline PC model.

3.3.1. Inference

Marginalization: To handle missing data, the PC marginalizes over the unobserved class variable Y by summing out the probabilities across all possible class labels. This is achieved using imputation techniques, where the missing values in the input are replaced with their most probable values based on the learned distributions in the PC. The marginal probability of the observed features $\mathbf{X}_{\text{obs}}$ is computed as:

$$P(\mathbf{X}_{\text{obs}}) = \sum_{Y} P(\mathbf{X}_{\text{obs}}, Y)$$

where $P(\mathbf{X}_{\text{obs}}, Y)$ is the joint probability computed by the circuit. In the code, this is implemented through the function `tp.utils.impute.impute`, which uses the PC model to infer the missing values. For instance:

- The CNN-extracted features X_{mixed} are combined with an unknown label $Y = \texttt{nan}$ to form a concatenated input tensor.
- The `b_impute` function applies batch imputation by marginalizing over the unknown variable Y, effectively reconstructing the missing values based on the PC's probabilistic structure.

Conditional Probability: The PC computes the conditional probability of the class label Y given the observed features $\mathbf{X}_{\text{obs}}$. This is expressed as:

$$P(Y \mid \mathbf{X}_{\text{obs}}) = \frac{P(\mathbf{X}_{\text{obs}}, Y)}{P(\mathbf{X}_{\text{obs}})}$$

Here, $P(\mathbf{X}_{\text{obs}}, Y)$ is the joint likelihood of the features and class label, while $P(\mathbf{X}_{\text{obs}})$ is the marginal likelihood of the features. The function `tp.utils.impute.b_impute` computes this by iteratively applying the PC's weighted summation and normalization steps. In the code, the inferred class labels are obtained by evaluating the imputed output:

- The imputed values from the PC include the predicted class label Y_{pred}, which corresponds to the most probable class for the given features.
- The number of correctly predicted labels is compared against the true labels Y_{test} to compute the model's accuracy.

4. METHODS

We wanted to check if the Mixed CNN-PC model could be used to reach a higher classification accuracy than a pure PC model. Our first experiment focused on comparing the performance of our Mixed model against a Baseline PC. As an additional verification we compared the Mixed model to the CNN model. In two follow-up experiments we wanted to see if we can leverage the nature of the PC in tasks which typically benefit from using probabilistic reasoning: a) training on small datasets, b) handling missing data.

4.1. Toolbox

We used the TNSPC toolbox provided by the project's supervisor Niccolò Ciolli, which leverages the structure and functionality of PCs with PyTorch for efficient probabilistic reasoning. The toolbox includes a wide range of utilities for constructing and training PC models, supporting various types of leaf nodes and operations. The code for the models is available in this GitHub repository.

4.2. Accuracy comparison

We compared the accuracy of three models. We trained all models on 8000 images from the MNIST dataset. All training was done in 10 epochs for each model. We trained the Baseline PC and the CNN model on the whole images sized 28x28. The Mixed CNN-PC model was trained on the convolution outputs extracted from the CNN model. Importantly, the number of parameters was very similar for the Baseline and Mixed CNN-PC models ($\sim$ 8000). The CNN model had a higher number of parameters in order to ensure robust convolutions. We tested the accuracy of our models on 2000 images from the dataset that were not presented during training.

4.3. Experiment 1 - Small training size

We wanted to check if the Mixed CNN-PC models' probabilistic nature allows for better predictions when trained on a small training dataset. We compared the accuracies of our Mixed CNN-PC model and the Feed Forward part of the CNN model (FFN). As input we used the convolution outputs extracted from the training of the CNN Model. We experimented with training the models on datasets consisting of: [10, 100, 1000] samples.

4.4. Experiment 2 - Missing data

In Experiment 2 we wanted to test the Mixed CNN-PC model's ability to handle missing data. Again, we used the FFN for comparing the results. We iteratively removed parts of data of different sizes. We compared the models by removing: [4, 16, 32, 64] out of the flattened input vector of length 96.

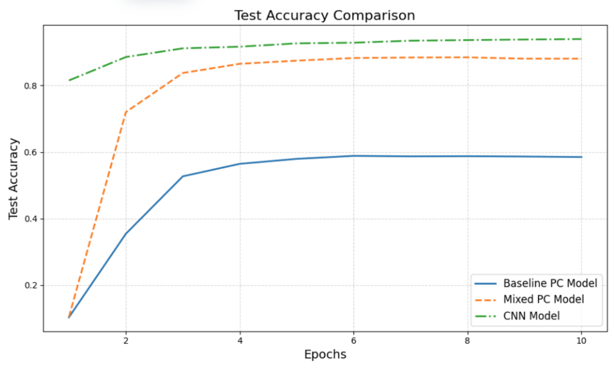

Fig. 2. Accuracy comparison of the Baseline PC model (blue), Mixed CNN-PC model (orange) and CNN model (green) throughout 10 epochs.

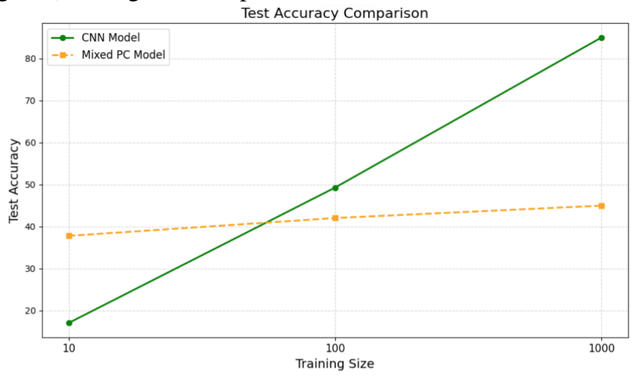

Fig. 3. Experiment 1 - accuracies of the FFN model (green) and Mixed CNN-PC model (orange) for different sample size.

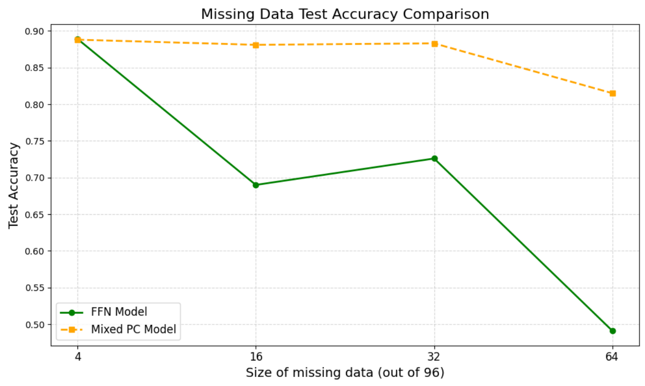

Fig. 4. Experiment 2 - accuracies of the FFN model (green) and Mixed CNN-PC model (orange) for handling missing data.

5. RESULTS

5.1. Accuracy Comparison

Figure 2 shows the test accuracies of the three models over 10 epochs. The CNN achieved the highest accuracy (0.94), followed by the Mixed CNN-PC model (0.88), which outperformed the Baseline PC model (0.58). This demonstrates that integrating CNNs with PCs improves performance over standalone PCs, though it slightly lags behind the CNN.

5.2. Experiment 1 - Small Training Size

As shown in Figure 3, the Mixed model outperformed the CNN model when trained on small datasets, achieving better stability and higher relative accuracy with limited training data.The CNN model demonstrated a rapid improvement in accuracy with increased sample size reaching 85% for 1000 samples, while the Mixed model's growth slowed. This highlights the Mixed model's ability to generalize effectively with limited data due to probabilistic reasoning capabilities.

5.3. Experiment 2 - Missing Data

Figure 4 illustrates the robustness of the models to missing data. The Mixed CNN-PC model achieved significantly higher accuracy (81.50%) than the CNN (49.10%), demonstrating its superior handling of uncertainty and incomplete inputs, enabled by the probabilistic reasoning of its PC layers.

6. DISCUSSION

In the project we show a successful implementation of a Mixed model, consisting of convolutional and probabilistic layers. In the accuracy comparison we showed that this novel architecture is capable of reaching a much higher accuracy than the pure PC baseline model. The accuracy wasn't as high as the accuracy of the pure CNN model, however this was expected due to the larger number of parameters in the latter. Notably, the Baseline model was trained directly on whole MNIST images, while the Mixed model utilized convolutional features extracted by the CNN. This highlights a key distinction: the Mixed model applies probabilistic reasoning not to raw images but to convolutional outputs, simplifying image representations. This approach could reduce dataset size by storing convolutional outputs for subsequent probabilistic tasks, such as decision-making under uncertainty.

Experiments 1 & 2 showed that our implementation was able to leverage the advantages of probabilistic modeling. We showed that the Mixed model performed better when trained on a very small input size (10). This effect however disappeared for training inputs larger than 100. We also showed that the Mixed CNN-PC model reached an accuracy much higher than the CNN model when tested on missing data.

An interesting follow-up could be to look at the Mixed model's approach as resembling the human sensory information processing. Since it has been shown that: a) the pathways in visual cortex follow a convolutional structure [4] and b) humans often have to perform probabilistic reasoning to make decisions in settings of uncertainty [5], the Mixed model could be used as a model of these processes.

7. REFERENCES

[1] H. Poon and P. Domingos, "Sum-product networks: A new deep architecture," in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, 2011, pp. 689–690.

[2] R. Peharz, A. Vergari, K. Stelzner, A. Molina, X. Shao, M. Trapp, K. Kersting, and Z. Ghahramani, "Random sum-product networks: A simple and effective approach to probabilistic deep learning," in *Proceedings of the 35th Uncertainty in Artificial Intelligence Conference*. Proceedings of Machine Learning Research, 2020.

[3] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou, "A survey of convolutional neural networks: analysis, applications, and prospects," *IEEE transactions on neural networks and learning systems*, vol. 33, no. 12, pp. 6999–7019, 2021.

[4] Pulkit Agrawal, Dustin Stansbury, Jitendra Malik, and Jack Gallant, "Convolutional neural networks mimic the hierarchy of visual representations in the human brain," *arXiv preprint arXiv:1406.3284*, 2014.

[5] E. K. Hokor, "Probabilistic thinking for life: The decision-making ability of professionals in uncertain situations," *International Journal of Studies in Education and Science (IJSES)*, vol. 4, no. 1, pp. 31–54, 2023.