Slide, Project, Learn, Repeat

Jiri Pivrnec

August 21, 2024

Abstract

This paper introduces a novel neural network layer, **ProjConv2D**, designed as an alternative or complementary module to the standard convolutional layer in Convolutional Neural Networks (CNNs). The ProjConv2D layer leverages a learnable projection matrix to transform sliding patches of the input image, offering a different mechanism for feature extraction. Experiments conducted on FashionMNIST and CIFAR-10 datasets demonstrate that the ProjConv2D layer, particularly when used as the first layer in a CNN, improves learning efficiency and performance compared to traditional CNN architectures. Additionally, a comparative analysis with Vision Transformers (ViTs) highlights the similarities and differences in the underlying mechanisms. We present the mathematical formulation of the ProjConv2D layer, detail the experimental setup, and discuss the implications of our findings.

1 Introduction

Convolutional Neural Networks (CNNs) have become the cornerstone of modern computer vision tasks due to their ability to efficiently capture spatial hierarchies in images. Traditional convolutional layers operate by sliding a filter across the input and learning feature maps through a series of linear transformations followed by non-linear activations [1]. Recently, Vision Transformers (ViTs) have emerged as an alternative approach, leveraging self-attention mechanisms over patches of images [2].

This paper proposes a novel layer, **ProjConv2D**, which combines the sliding window operation of CNNs with a learnable projection step. This layer offers an alternative to the traditional convolutional layer, potentially enhancing feature extraction by projecting patches into a new feature space before further processing. We present a mathematical formulation of ProjConv2D, compare it with existing methods like CNNs and ViTs, and evaluate its performance on standard image datasets.

2 ProjConv2D: Definition and Formulation

2.1 Mathematical Formulation

The ProjConv2D layer operates by first unfolding the input tensor x into patches. Given an input of shape (B, C_{in}, H, W) where B is the batch size, C_{in} is the number of input channels, and H, W are the height and width, the ProjConv2D layer slides over the input with a kernel of size $K \times K$, producing patches of size $C_{\text{in}} \times K \times K$.

Let x_{unfolded} represent these patches, reshaped to $(B, P, C_{\text{in}} \times K \times K)$ where P is the number of patches. The ProjConv2D layer learns a projection matrix W of size $C_{\text{out}} \times (C_{\text{in}} \times K \times K)$ that transforms each patch into a new feature space:

$$y_{\text{projected}} = W \cdot x_{\text{unfolded}}$$

This operation results in a tensor of shape (B, C_{out}, P), which is then reshaped to $(B, C_{\text{out}}, H', W')$ where H' and W' depend on the stride and padding.

The ProjConv2D layer can be formulated as:

$$y = \text{reshape}(W \cdot \text{unfold}(x))$$

where the projection matrix W is the primary learnable parameter of the layer.

2.2 Implementation

The implementation of the ProjConv2D layer in PyTorch is as follows:

```
class ProjConv2D(nn.Module):
    def __init__(self, in_channels, out_channels, kernel_size, stride=1, padding=0):
        super(ProjConv2D, self).__init__()
        self.projection_matrix = nn.Parameter(torch.randn(out_channels, in_channels * kernel

    def forward(self, x):
        x_unfolded = nn.functional.unfold(x, kernel_size=self.kernel_size, stride=self.stric
        projected = torch.matmul(self.projection_matrix, x_unfolded)
        batch_size, out_channels, num_patches = projected.size()
        new_height = int((x.size(2) + 2 * self.padding - self.kernel_size) / self.stride + 1
        new_width = int((x.size(3) + 2 * self.padding - self.kernel_size) / self.stride + 1)
        projected = projected.view(batch_size, out_channels, new_height, new_width)
        return projected
```

3 Comparison with Vision Transformers (ViTs)

Vision Transformers (ViTs) have garnered attention as an alternative to CNNs by applying the transformer architecture, traditionally used in NLP, to image

data. This section compares the ProjConv2D layer with the architecture of ViTs, focusing on their mathematical formulations and operational differences.

3.1 Patch Extraction

Both ProjConv2D and ViTs start by breaking down the input image into patches. However, the nature of these patches and the subsequent operations differ:

- **ProjConv2D Layer**: Extracts overlapping patches using a sliding window approach, similar to the convolution operation. These patches are transformed using a learnable projection matrix.
- **ViTs**: Non-overlapping patches of a fixed size are extracted from the input image. Each patch is then flattened and linearly projected into an embedding space before being passed into a series of transformer layers.

Mathematically:

- ProjConv2D patch extraction can be represented as:

$$\text{ProjConv2D: } x_{\text{unfolded}} = \text{unfold}(x)$$

  where x is the input tensor.

- ViT patch extraction is:

$$\text{ViT: } x_{\text{patch}} = \text{reshape}(\text{patches}(x))$$

  where patches are non-overlapping and of fixed size.

3.2 Projection and Embedding

In the ProjConv2D layer, each extracted patch is projected into a new feature space using a learnable matrix. This projection is similar to the linear embedding in ViTs, but the mechanism is different:

- **ProjConv2D**: Applies a learned projection matrix to each patch independently, resulting in a transformed feature map.

$$y_{\text{projected}} = W \cdot x_{\text{unfolded}}$$

- **ViTs**: Linearly embed each flattened patch into a higher-dimensional space, often followed by the addition of positional encodings.

$$z_0 = [x^1_{\text{patch}}E; x^2_{\text{patch}}E; \ldots; x^N_{\text{patch}}E] + E_{\text{pos}}$$

  where E is the embedding matrix and E_{pos} is the positional encoding.

3.3 Self-Attention vs. Localized Projection

- **ViTs**: Use self-attention mechanisms to capture long-range dependencies across all patches in an image. Each patch can attend to every other patch, which allows for global context but increases computational complexity [4].

$$\text{Self-Attention: } \text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

  where Q, K, V are query, key, and value matrices.

- **ProjConv2D**: Focuses on localized feature extraction within each sliding patch. The operation is more akin to convolution, emphasizing spatial locality, which may limit the global context understanding but maintains efficiency.

3.4 Positional Encoding

- **ViTs**: Employ positional encodings to maintain spatial information after patch flattening and linear embedding. This is crucial since the transformer architecture itself does not inherently capture the spatial structure of the input [2].

$$z_0 = z_0 + E_{\text{pos}}$$

- **ProjConv2D**: Retains spatial structure inherently due to the nature of the sliding window and the way feature maps are reconstructed after projection. There is no need for additional positional encodings as the spatial arrangement is preserved by design.

4 Experimental Setup and Results

4.1 Experimental Setup

The ProjConv2D layer was integrated into a CNN architecture, and its performance was evaluated on FashionMNIST and CIFAR-10 datasets. The network configurations tested include:

- **ProjConv2D as the first layer followed by traditional convolutional layers**.
- **All layers as ProjConv2D**.
- **All layers as standard convolutional layers** (baseline).

The models were trained for 5 epochs with identical hyperparameters for fair comparison.

4.2 Results

The results from the experiments on the FashionMNIST dataset are summarized below:

Model Configuration	Validation Accuracy	F1-Score	Training Time (seconds
ProjConv2D (1st Layer) + Conv Layers	90.97%	0.9100	19.00
All ProjConv2D Layers	88.92%	0.8896	25.14
All Conv Layers (Baseline)	90.66%	0.9065	19.37

Table 1: Results on FashionMNIST Dataset

5 Discussion

5.1 ProjConv2D vs. ViTs

The experiments suggest that while the ProjConv2D layer is less effective when used in all layers, it significantly enhances the feature extraction capability when used as the first layer in conjunction with traditional convolutional layers. In comparison to ViTs, ProjConv2D offers a middle ground between the localized feature extraction of CNNs and the global context modeling of transformers. ProjConv2D's use of a learnable projection matrix provides a more flexible transformation compared to the fixed filters in CNNs, yet it retains the spatial locality that is often lost in ViTs.

5.2 Computational Efficiency

ViTs, despite their success, often require large amounts of data and computational resources due to the quadratic complexity of the self-attention mechanism [4]. ProjConv2D, in contrast, maintains computational efficiency by operating in a localized manner and may serve as a practical alternative or complement in resource-constrained environments.

6 Conclusion

This paper introduces the ProjConv2D layer, a novel approach to feature extraction in CNNs. The ProjConv2D layer shows promise in improving model performance when used strategically within a network. In comparison to ViTs, ProjConv2D retains spatial locality while offering a flexible transformation mechanism. Future work will explore different configurations and deeper network architectures to fully understand the potential of ProjConv2D in various computer vision tasks.

Acknowledgements

I would like to thank for a peer review and helpful comments.

References

References

[1] LeCun, Y., et al., "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, 1998.

[2] Dosovitskiy, A., et al., "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," *ICLR*, 2021.

[3] Howard, A. G., et al., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," arXiv preprint arXiv:1704.04861, 2017.

[4] Vaswani, A., et al., "Attention is All You Need," *NeurIPS*, 2017.

[5] Simonyan, K., & Zisserman, A., "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv preprint arXiv:1409.1556, 2014.