Mind Your Own Kernels: Dynamic Convolution for Personalized Feature Extraction

A PREPRINT

Anmol Sharma
A machine learning enthusiast
Punjab,India

DATE

ABSTRACT

Convolutional Neural Networks (CNNs) have been widely used for image classification tasks, but they often require large amounts of storage and computation due to the fixed set of learned filters. In this paper, we propose a novel approach to address this issue by introducing Dynamic Convolutional Neural Networks (DCNNs). Our DCNNs learn a set of custom kernels for each patch in the input image, which allows for more efficient computation and reduced storage requirements. The learned kernels are specific to the input image, making our approach partially inspired by the human brain's ability to recognize patterns based on input.

1 Introduction

Convolutional Neural Networks (CNNs) have achieved remarkable success in various computer vision tasks such as image classification, object detection, and segmentation. The Conv2d layer is a fundamental building block of CNNs, which applies a set of fixed filters (kernels) to extract features from the input image. However, storing and computing these kernels can be computationally expensive and memory-intensive, especially for large images or complex architectures. To address this issue, various techniques such as depthwise separable convolutions, dilated convolutions, and group convolutions have been proposed to reduce the number of parameters and computations. However, these methods still require storing a large number of predefined kernels.

In this paper, we propose a custom Convolution layer, where kernels are predicted dynamically using a hybrid dense layer for each patch of the input image. This approach saves time and storage as the model doesn't have to remember kernels. The proposed method learns to predict the kernels that best extract features from the input patch, based on the patch's content. Our approach is inspired by recent works on dynamic convolution, which has shown promising results in various tasks such as object detection and segmentation.

2 Related Work

Pixel-Adaptive convolutional networks:

Pixel-Adaptive Convolutional Networks (PAC) is a content-adaptive convolution layer that addresses limitations of existing content-adaptive layers while retaining the advantages of spatially invariant convolution. In PAC, a standard spatially invariant convolution filter W is multiplied with a spatially varying filter K, known as an "adapting kernel," at each pixel. This adapting kernel has a pre-defined form, such as Gaussian or Laplacian, and is dependent on the pixel features, referred to as "adapting features." These features can be pre-defined, such as pixel position and color features, or can be learned using a CNN. However, learning all aspects of the input using the adapting features can be memory-intensive. To address this issue, our Hybrid layer confines each patch size as a learnable area. This helps in reducing the number of parameters and memory requirements while retaining the desirable properties of content-adaptive filtering.

Dynamic filter networks:

Dynamic Filter Networks (DFN) are a type of content-adaptive filtering technique where filter weights are predicted by a separate network branch to provide customized filters for different input data. Wu et al. extended this work by adding an attention mechanism and a dynamic sampling strategy, allowing position-specific kernels to learn from neighboring regions. This helps in capturing more contextual information from the input. However, predicting all position-specific filter weights requires many parameters, making it challenging to scale to multiple dynamic-filter layers. Careful architecture design is necessary to address this issue. DFN has been applied to various task-specific use cases, such as motion prediction, semantic segmentation, and Monte Carlo rendering denoising.

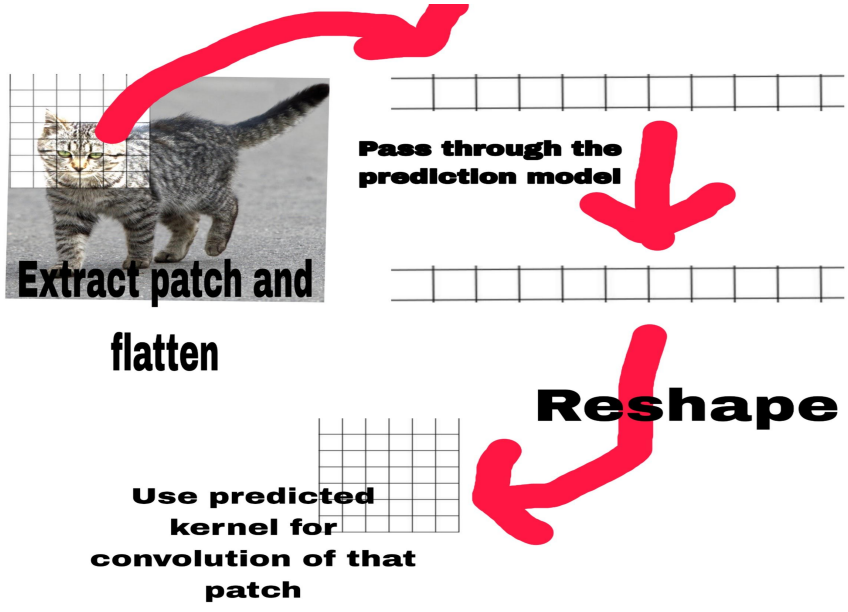
3 Inspiration and working

The DynamicConv layer is a novel convolutional neural network layer that is designed to dynamically learn and apply convolutional filters on input data. The layer is inspired by the way the human brain processes visual information, as it is known that the patterns that a person finds in their daily life are dependent upon what they are looking at. In this sense, the DynamicConv layer is able to adapt to the input data, and can learn to extract meaningful features from it. The main advantage of the DynamicConv layer is that it allows for the learning of custom convolutional filters for each patch of the input data. This means that rather than applying a fixed set of memorized filters to every patch, the layer can learn filters that are specific to each patch, which can greatly reduce the amount of computation and storage required for the network.

3.1 Technique

Our proposed custom Conv2d class is designed to predict kernels dynamically for each patch of the input image. Specifically, given an input patch of size HxWxC, we use N hybrid dense layers to predict a set of kernels of size KxKxC (and then concatenate them), where K is the kernel size, C is the number of input channels, and N is the number of output channels. This combination allows us to reduce the number of parameters and computations compared to a standard dense layer.

The predicted kernels are then applied to the input patch using the standard convolution operation to obtain the output feature map of size H'xW'xN. The predicted kernels are not stored explicitly, but are computed on-the-fly for each patch. The weights of the hybrid dense layer are learned during training using backpropagation and gradient descent. During inference, the hybrid dense layer is used to predict the kernels for each patch, and the convolution operation is applied to obtain the output feature map.



3.1.0 The Hybrid dense layer

The HybridDense layer is a custom neural network layer designed for use in deep learning models. This layer is based on the traditional dense layer, but with additional

modifications to the activation function and scaling parameters. The layer takes input data and performs a linear transformation on it using a weight matrix and bias vector. The output of this linear transformation is then passed through a rectified linear unit (ReLU) activation function with a small epsilon value added to prevent numerical instability. Next, the output is multiplied by a set of learned scaling parameters and raised to learned powers, which can be clipped to prevent numerical issues. Finally, the sign of the linear output is applied to the result, and any NaN values are replaced with zero. This layer can be used in deep learning models to help improve performance and accuracy by providing a more flexible and customizable activation function. It has the potential to be especially useful in applications where non-linear relationships between input and output variables are complex and difficult to model using traditional activation functions.

3.1.1 The Hybrid Convolutional layer.

The DynamicConv layer is a custom neural network layer designed for use in deep learning models. This layer is based on the traditional convolutional layer, but with additional modifications to allow for more dynamic and adaptive feature learning. The layer takes input data and extracts image patches using a specified patch size, stride, and padding. It then uses a set of learned predictors, which are modeled as a series of hybrid dense layers, to predict a kernel for each patch. These kernels are then used to perform convolution on the input data, resulting in a set of feature maps. The layer also includes a similar_conv layer, which is a traditional convolutional layer used to learn the most common features. The outputs from both the dynamic convolution and similar convolution are added together to produce the final output. One key advantage of the DynamicConv layer is that it can learn more complex and adaptive features than traditional convolutional layers, which typically use a fixed set of kernels. This can help improve the accuracy and generalization of deep learning models, especially in applications where the relationships between input and output variables are complex and difficult to model using traditional convolutional layers.

3.2 The kernel prediction models

For n output dimensions, n kernel prediction models are used, which are not connected to each other. Each of them accepts [patch_size,patch_size,input channels] patch as input and returns a kernel of the same size. This kernel is then multiplied with the patch and summated to get the final convolved output. The outputs are concatenated to get the final result.

4 Impact

The proposed custom Convolution layer with dynamic kernel prediction has the potential to significantly impact on-device computing and edge hardware. By dynamically predicting kernels for each patch of the input image, this approach can significantly reduce the memory and computation requirements of convolutional neural networks (CNNs). This reduction in memory and computation is particularly important for edge devices and mobile platforms with limited resources. Since the proposed method eliminates the need to store pre-defined kernels, it reduces the memory footprint of the CNN model. Additionally, by predicting kernels on the fly, the proposed method can reduce the computational cost of convolution operations, making it more suitable for deployment on edge devices and mobile platforms with limited computing resources. In summary, the proposed custom Convolution layer with dynamic kernel prediction has the potential to enable more efficient deployment of CNNs on edge devices and mobile platforms, which can have a significant impact on a wide range of applications such as autonomous vehicles, mobile robotics, and augmented reality.

5 Experiments

4.1 Autoencoder

We created an autoencoder of input size 512x512x3 and hidden size 64x64x4.
The encoder consists of a single DynamicConv layer that reduces the spatial dimension of the input by a factor of 8. The decoder, on the other hand, has two layers: a DynamicConv layer that increases the spatial dimensions of the input by a factor of 8 and a PixelShuffle layer to convert the channel dimension to the original 3 RGB channels. The output of the decoder is clipped to a range of 0 to 1 using the ReLU activation function.
This implementation serves as an experiment to explore the effectiveness of the DynamicConv layer in an image compression application using an autoencoder architecture.The total model size is 1.79 mb ,with encoder size being only 89 kb.These were trained for ~1 hr on a P100 gpu

Inference time: ~.0.52 seconds on a cpu

Below are the results:

Decoded

Ground Truth

All the code is available at:

Keep-up-sharma/Faster-and-More-efficient-hybrid-layers (github.com)

Conclusion:

In this paper, we have proposed a novel approach for personalized feature extraction in convolutional neural networks. Our Dynamic Convolutional Neural Networks (DCNNs) dynamically learn custom kernels for each patch of the input image, allowing for more efficient computation and reduced storage requirements. Our approach is inspired by recent works on dynamic convolution, but with modifications to reduce the number of parameters and computations. We believe that our proposed approach can be applied to various computer vision tasks such as image classification, object detection, and segmentation, and can potentially lead to better performance and faster inference times. Future work can explore further optimizations and extensions of our proposed approach**.**