

CNN

Name: Usama Liaqat

ID: 24081723

Github Repo: [usamakund/Machine-Learning-and-Neural-Networks](https://github.com/usamakund/Machine-Learning-and-Neural-Networks)

Convolutional Neural Networks: Foundations and Relevance

Cell CNNs have been ubiquitous in the current state of machine learning, mostly in computer vision applications including image classification, object detection and semantic segmentation. Contrary to a more basic fully connected net that analyzes each pixel individually as a single entity, CNNs utilize the natural melodies of images through the application of a set of learnable filters/kernels on local pixel blocks and derives saliency features (e.g. edges, textures, patterns, etc.) (Gupta et al., 2022). Inside a convolutional layer, each filter moves through the input image volume or feature map, and calculates dot products at each spatial location to generate an activation map, sometimes also known as a feature map, which highlights the existence and strength of a specific learned feature in different locations in the input. These learned filters can be simple edge detectors, color blobs, and as one goes deeper into the network, can become more and more abstract. CNNs are inspired by conceptually the biological visual systems where neurons react to local receptive locations and possess the quality of translational invariance: the quality that allows the network to identify patterns regardless of their position in the image.

Filter size (dimension of a kernel), stride (step size of sliding), and padding (border treatment) choice has a direct effect on the ability of a CNN to capture a feature at various scales, control spatial resolution, as well as preserve, a boundary signal. To come up with effective networks, there is a need to clearly understand the influence of these hyperparameters on the process of feature extraction (Purwono et al., 2022). Though CNNs are indeed trained to learn filters, the structural decisions, including the size of the filters, are determined by the model designer and therefore can have a significant influence on behavior of the model, the amount of computations required, and the representational capability of the model.

Why Filter Size Matters: 3×3 vs. 5×5

Filter size defines the spatial receptive field of the convolutional layer that is, the number of neighboring pixels that the network will take into account in a single extraction of features. A 3x3 filter looks at a small patch of 3 pixels x 3 pixels thus picking fine local patterns like elementary edges and textures. A 5x5 filter, on the other hand, has a larger 5x pixel x 5x pixel field of view, which is potentially more beneficial in finding more complicated or lower-level structures in an image but at a higher parameter expense and cost of computation. One architectural rule that has

been relied on in deep learning is a general rule of using consistently 3×3 filters in effective networks like VGGNet. VGGNet showed that a pair of 3×3 convolutional layers provide a receptive field just as good as a 5×5 convolution and requires fewer parameters and an increase in the network depth, enhancing the learning capacity and nonlinear representation. The use of a series of small filters adds additional non-linearities (as in forms of ReLU operations between layers) thus increasing the capacity to model complex functions by the network (Zha et al., 2022). Empirical and empirical studies on the effects of a filtersize indicate more complex models may be constructed using larger kernels; however, larger kernels require more powerful computational resources and assumed training data to help prevent overfitting which is harmful to model performance on some datasets. Where filter size was manipulated and other parameters held constant in controlled experiments, a performance enhancement with the selected filter size was sometimes observed and is thought to introduce smaller receptive fields and greater depths of architecture is highly likely to give a more balanced trade-off between feature extraction and generalization. This duo of endorse and aggregate capture (3×3) and larger context capture (5×5) acts on the wafer in which the initial layers obtain the low-level examples of the data set, and the ensuing layers influence the accumulation of the initial layer knowledge by the later layers into the central concepts. As an example, first convolutional layers that have small kernels identify edges and simple forms, but more deeply, multiple features are combined and create complex portions of objects. Larger kernels on the other hand can attenuate small features and highlight coarse features that can be unsatisfactory in a task where fine boundaries or texture are needed (Bhatt et al., 2021).

Stride and Its Impact on Feature Resolution

Stride determines the size of how far the filter moves across the width and height in the input feature map. One stride implies that the filter will be moved pixel by pixel and thus have a solid coverage of the image, and create larger feature maps which retain spatial detail in them. Strides longer than two reduce the spatial dimensions of the feature map since the filter omits intermediate pixels meaning that the representation is down-sampled. This minimization may prove beneficial in countering model size, memory usage and calculational need; however, it also compromises tiny spatial detail of the ensuing feature maps (Zha et al., 2022). The convolutional operation

output size in a convolutional operation with a stride of size s may be determined mathematically as:

$$Output\ Size = \lceil \frac{N + 2P - F}{\delta} \rceil + 1$$

Strong stride views the output as aggressively reducing the output resolution but at the same time increasing the effective receptive field of each activation as the unit is summing up information over much more distant input pixels. In terms of feature extraction, strides can be larger to produce less refined representations which can potentially lose a subtle pattern in the data. However, such a tradeoff could be explained by a faster training and reduction of memory usage, especially with deeper networks where concerted pooling layers or successive strides must make spatial reduction hierarchical (Gupta et al., 2022).

Padding: Preserving Edge Information

In the case of filtering the input image with no padding (so-called operation is often called a valid convolution), the final feature map is smaller than the input, particularly at the edges. Though this dimensional contraction may result in the under-utilisation or loss of information at the edge positions in images as edge pixels are part of fewer receptive fields when performing a convolution. Padding This is normally achieved by adding zeros (zero-padding) to the boundary of the input and enlarges rows and columns of pixels around the input to maintain their spatial dimensions and to ensure that each original pixel, even those in the periphery, is contributed equally to the feature extraction process (Gupta et al., 2022). When the stride is one the size of the output feature map is the same as the size of the input with same padding. Padding is also ensured to ensure that deeper networks do not overly smooth out representations of space nor do filters cover the edge regions appropriately as they cover the central regions. Padding produces a relatively small cost of computation, but ensures consistency of features maps and better ability to accurately learn edge features which can be essential in the initial convolutional layers.

Design Insights and Practical Considerations

The choice of filter sizes, stride variations, and padding conventions in modern convolutional neural network (CNN) architectures represent a traded off compromise between the convergence of feature representation and computability. Small filters like the popular 3x3 filter are typically

used where the task requires a high level of spatial resolution and where the model depth can be increased without an excessive consumption of resources. The small kernels produce dense activation maps and form the basis of hierarchical feature extraction that is synonymous with deep CNN architecture. On the other hand, larger kernels can be placed in special layers at the top to hasten localization of global context, especially in case of reduced spatial resolution is tolerable. Stride and padding devices are additive to these options and have the effect of space resolution modulation, edge behaviour control and influence of parametric load. Practically, researchers often start with existing backbones e.g. VGG, ResNet, or modern variants; they use these models as baseline before drawing the filter sizes, strides and padding setups to the details of the dataset, feature complexity, and hardware constraints (Purwono et al., 2022).

Original Input Image



Code Demonstration: Visualizing Impact of Filter Size

The Python script that should be run inside a Jupyter notebook demonstrates the effects of filter size, stride, and padding variation in an analysis of both learned feature maps of CIFAR-10 images. It uses TensorFlow/Keras to build exemplar models, as well as to create visualizations of the activation maps obtained with respect to one test image.

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt

from tensorflow.keras.datasets import cifar10
from tensorflow.keras import models, layers

# Load a sample from CIFAR-10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
x_sample = x_test[0:1] / 255.0

def build_model(kernel_size, stride, padding):
    model = models.Sequential([
        layers.Conv2D(16, kernel_size, strides=stride, padding=padding, activation='relu',
```

```

        input_shape=(32, 32, 3)),
    ])
    return model

# Define configurations
configs = [
    (3, 1, 'same'),
    (3, 2, 'same'),
    (5, 1, 'same'),
    (5, 1, 'valid'),
]

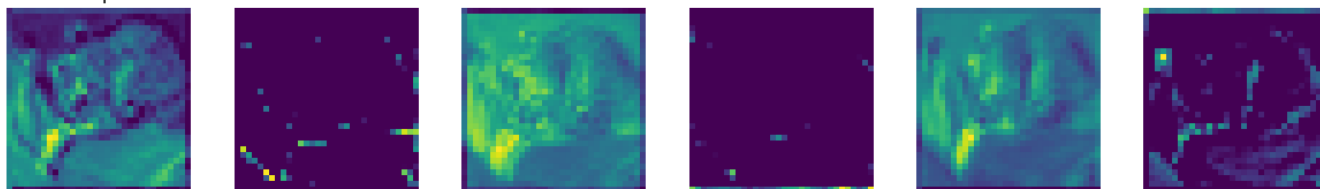
fig, axes = plt.subplots(len(configs), 6, figsize=(12, 8))
for i, (k, s, p) in enumerate(configs):
    model = build_model((k, k), s, p)
    activations = model.predict(x_sample)
    for j in range(6):
        axes[i, j].imshow(activations[0, :, :, j], cmap='viridis')
        axes[i, j].axis('off')
    axes[i, 0].set_title(f'{k}x{k} s={s} p={p}')

plt.tight_layout()
plt.show()

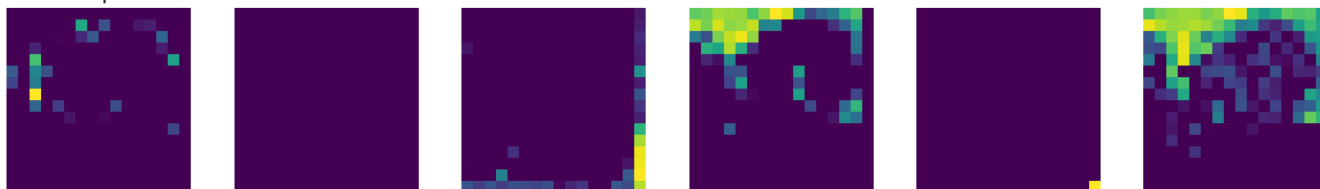
```

The script generated four variants of CNNs with different kernel size (3 3 versus 5 5), stride and padding. The initial six feature maps resulting from the convolutional layer are visualised in a CIFAR-10 test sample image in each configuration. With these visualisations, learners are able to see the impact of hyperparameter change on feature patterns, resolution, and the level of abstraction in the extracted representations.

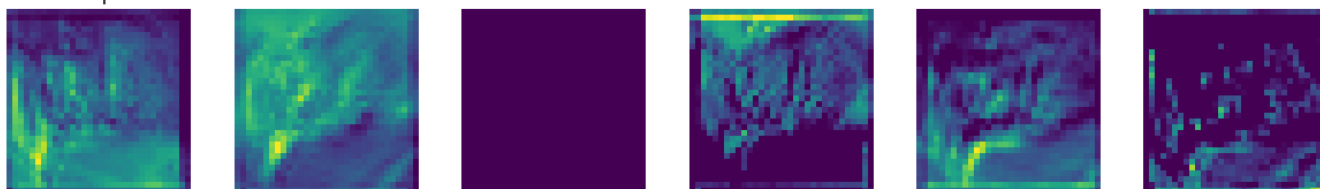
3x3 s=1 p=same



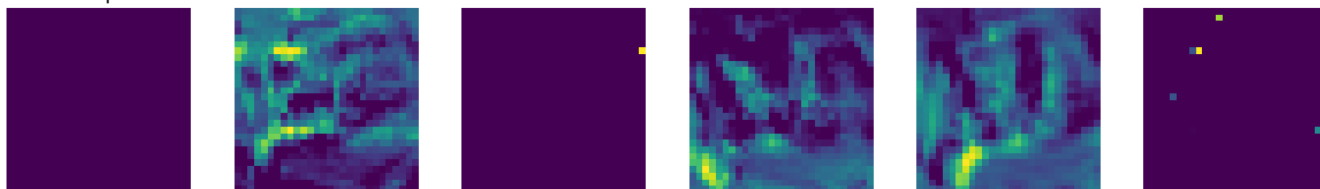
3x3 s=2 p=same



5x5 s=1 p=same



5x5 s=1 p=valid



References

- Bhatt, D., Patel, C., Talsania, H., Patel, J., Vaghela, R., Pandya, S., Modi, K. and Ghayvat, H., 2021. CNN variants for computer vision: History, architecture, application, challenges and future scope. *Electronics*, 10(20), p.2470.
- Gupta, J., Pathak, S. and Kumar, G., 2022, May. Deep learning (CNN) and transfer learning: a review. In *Journal of Physics: Conference Series* (Vol. 2273, No. 1, p. 012029). IOP Publishing.
- Purwono, P., Ma'arif, A., Rahmانيar, W., Fathurrahman, H.I.K., Frisky, A.Z.K. and ul Haq, Q.M., 2022. Understanding of convolutional neural network (cnn): A review. *International Journal of Robotics and Control Systems*, 2(4), pp.739-748.
- Zha, W., Liu, Y., Wan, Y., Luo, R., Li, D., Yang, S. and Xu, Y., 2022. Forecasting monthly gas field production based on the CNN-LSTM model. *Energy*, 260, p.124889.
- Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2017. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6), pp.84-90.
- He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).
- Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.