

Deep Learning Project

Classification of CIFAR-100 Dataset

Dataset Splitting:

- **Training Set:** The dataset is split into a training set and a test set. The training set consists of 50,000 images, with 500 images per class.
- **Test Set:** The test set contains the remaining 10,000 images, with 100 images per class.

Model Architecture:

```
def conv_block(in_channels, out_channels, pool=False):
    layers = [nn.Conv2d(in_channels, out_channels, kernel_size=3, padding=1),
              nn.BatchNorm2d(out_channels),
              nn.ReLU(inplace=True)]
    if pool: layers.append(nn.MaxPool2d(2))
    return nn.Sequential(*layers)
```

```
class ResNet9(ImageClassificationBase):
    def __init__(self, in_channels, num_classes):
        super().__init__()

        self.conv1 = conv_block(in_channels, 64)
        self.conv2 = conv_block(64, 128, pool=True)

        self.res1 = nn.Sequential(conv_block(128, 128), conv_block(128, 128))

        self.conv3 = conv_block(128, 256, pool=True)
        self.conv4 = conv_block(256, 512, pool=True)

        self.res2 = nn.Sequential(conv_block(512, 512), conv_block(512, 512))
        self.conv5 = conv_block(512, 1028, pool=True)
        self.res3 = nn.Sequential(conv_block(1028, 1028), conv_block(1028, 1028))

        self.classifier = nn.Sequential(nn.MaxPool2d(2), # 1028 x 1 x 1
                                         nn.Flatten(), # 1028
                                         nn.Linear(1028, num_classes)) # 1028 -> 100
```

The model is a simplified version of a Residual Network (ResNet) architecture, specifically a ResNet variant called ResNet9. ResNet architectures are known for their deep

layering with residual connections, enabling the training of very deep networks while mitigating the vanishing gradient problem.

Let's break down the structure of this ResNet9:

Structure:

1. Initial Convolutional Layers (conv1, conv2):

- **conv1:** Performs a convolutional operation with a 3x3 kernel, generating 64 output channels.
- **conv2:** Performs another convolutional operation, expanding to 128 channels, followed by batch normalization and ReLU activation. Additionally, it includes a max-pooling layer that reduces spatial dimensions by half (2x2) using a stride of 2.

2. Residual Blocks (res1, res2, res3):

- Each of these blocks represents a sequence of two basic blocks designed to maintain the network's depth while preserving learned features.
- Each basic block consists of two convolutional layers, each followed by batch normalization and ReLU activation. These blocks help learn more complex features.

3. Final Convolutional Layers (conv3, conv4, conv5):

- Similar to the initial layers, these perform convolutional operations, progressively increasing the number of channels (256, 512, 1028), followed by batch normalization and ReLU activation.
- Each of these layers also includes max-pooling to reduce spatial dimensions.

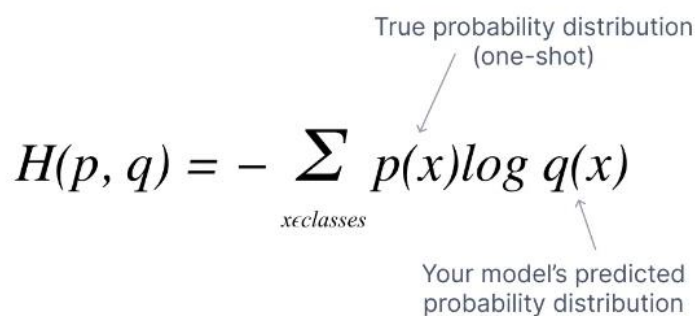
4. Classifier:

- **classifier** represents the final layers responsible for classification.
- It includes a max-pooling layer to further reduce spatial dimensions.
- The **Flatten** operation reshapes the tensor into a 1-dimensional tensor for further processing.
- Lastly, a fully connected (**Linear**) layer is used for classification, mapping the 1028-dimensional features to 100 classes (out_features=100).

Loss Used:

Cross Entropy loss was used for this multiclass classification.

$$H(p, q) = - \sum_{x \in \text{classes}} p(x) \log q(x)$$





Accuracy:

Test set results:

F1 score: 0.742467

Recall score: 0.743400

Accuracy score: 0.743400