TCNN on Images

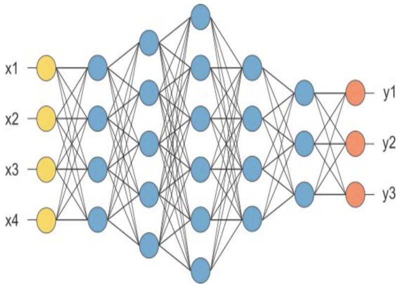
Zhiwang Yu

Department of mathematics
Southern University of Science and Technology

March 7th, 2024

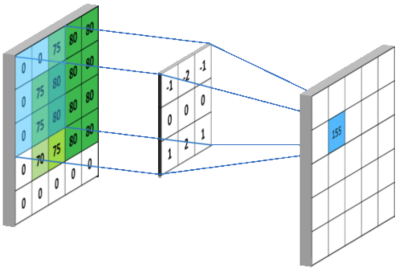
Contents

Feed Forward Neural Network



- Each circle represents a **node**.
- All nodes in one column construct a layer, where the layer with yellow color is the **input layer**, the layer with orange color is the **output layer** and the other layers are **hidden layers**.
- The edge in this graph represents the correspondence between the nodes.

Convolutional Kernel



- The left matrix is the data in the last layer.
- The middle matrix is the convolutional kernel.
- The right matirx is the data in the next layer.
- The data in the next layer is computed as the picture shows.

A Formula for Klein Bottle

A Formula for Klein Bottle

$$F_{\mathcal{K}}(\theta_1, \theta_2)(x, y) = \sin(\theta_2)(\cos(\theta_1)x + \sin(\theta_1)y) + \cos(\theta_2)Q(\cos(\theta_1)x + \sin(\theta_1)y),$$

where $Q(t) = 2t^2 - 1$. As given, $F_{\mathcal{K}}$ is a function on the torus, which is parameterized by the two angles θ_1 and θ_2. It actually defines a function on $\mathcal{K}$ since it satisfies $F_{\mathcal{K}}(\theta_1, \theta_2) = F_{\mathcal{K}}(\theta_1 + 2k\pi, \theta_2 + 2l\pi)$ and $F_{\mathcal{K}}(\theta_1 + \pi, -\theta_2) = F_{\mathcal{K}}(\theta_1, \theta_2)$.

Two Basic Kernels

-1	0	1
-1	0	1
-1	0	1

Table 1: The First Kernel of Klein Bottles

1	-1	1
1	-1	1
1	-1	1

Table 2: The Second Kernel of Klein Bottles

A Distance in Klein Bottle

The metric $d_{\mathcal{K}}$ is defined by

A Distance in Klein Bottle

$$d_{\mathcal{K}}(\kappa, \kappa') = \left(\int_{[-1,1]^2} \left(F_{\mathcal{K}}(\kappa)(x, y) - F_{\mathcal{K}}(\kappa')(x, y)\right)^2 dxdy \right)^{\frac{1}{2}}$$

for $\kappa, \kappa' \in \mathcal{K}$.

KF(CF) Layer

KF (CF) Layer

Let $M = S^1$ or $\mathcal{K}$ and let $\chi \subset M$ be a finite subset. Let $V_i = \mathbb{Z}^2$ and $V_{i+1} = X \times \mathbb{Z}^2$ be successive layers in a FFNN. Suppose V_{i+1} is a convolutional layer with threshold $s \geq 0$. Then V_{i+1} is called a **Circle Features (CF) layer** or a **Klein Features (KF) layer**, respectively, if the weights $\lambda_{-,(\kappa,-,-)}$ are given for $\kappa \in \chi$ by a convolution over V_i of the filter of size $(2s+1) \times (2s+1)$ with values

$$Filter(\kappa)(n,m) = \int_{-1+\frac{2m}{2s+1}}^{-1+\frac{2(m+1)}{2s+1}} \int_{-1+\frac{2n}{2s+1}}^{-1+\frac{2(n+1)}{2s+1}} F_M(\kappa)(x,y)\,dxdy$$

for integers $0 \leq n, m \leq 2s$.

The Construction of Kernels

- Let D be $[-1,1] \times [-1,1]$, consider the following function

$$f(x, y) = A + Bx + Cy + Dx^2 + Exy + Fy^2,$$

  then it becomes a 6-dimensional vector space $\mathcal{Q}$.
- Let $\int_D f = 0$ (mean centering) and $\int_D f^2 = 1$ (normalization). It is a 4-dimensional space $\mathcal{P} \subset \mathcal{Q}$.
- Let $f(x, y) = q(\lambda x + \mu y)$ with $\lambda^2 + \mu^2 = 1$ (rotation invariance). It implies that the space is 2-dimensional $\mathcal{P}_0 \subset \mathcal{P}$.

Kernels on a Klein Bottle

- Let $\int_{-1}^{1} q = 0$ and $\int_{-1}^{1} q^2 = 1$ with $q(t) = c_0 + c_1 t + c_2 t^2$, then $q \in A$, where A is homeomorphic to S^1. Let $\mathbf{v} = (\lambda, \mu)$, then $f(x, y) = q(\mathbf{v} \cdot (x, y))$. (Note that $\mathbf{v} \in S^1$.)
- Verify that $\int_D f = 0$ and $\int_D f^2 \neq 0$. Define a map $\theta : A \times S^1 \to \mathcal{P}_0$ by

$$\theta(q, \mathbf{v}) = \frac{q(\mathbf{v}, -)}{||f||_2}.$$

  θ is not a homeomorphism. Define a map $\rho : A \to A$ by

$$\rho(c_0 + c_1 t + c_2 t^2) = c_0 - c_1 t + c_2 t^2,$$

  then $\theta(q, \mathbf{v}) = \theta(\rho(q), -\mathbf{v})$.
- Verify that $\mathcal{P}_0$ is homeomorphic to a Klein bottle.

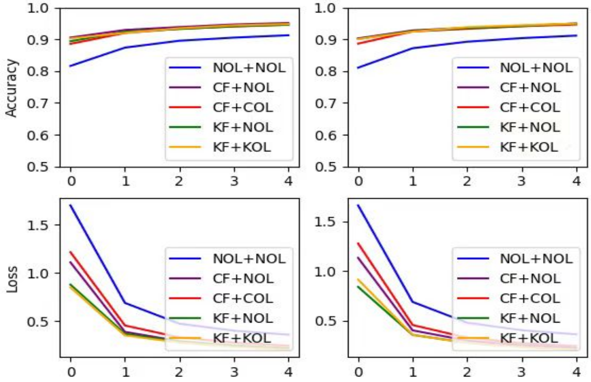
Main Program

```
img_input = Input(shape=(32, 32, 3))
net = Conv2D(64, (3,3), activation='relu', padding = 'same')(img_input)
#net = CircleFeatures(64)(img_input)
#net = KleinFeatures(64)(img_input)
net = MaxPooling2D(2,2)(net)
net = Conv2D(64, (3,3), activation='relu', padding = 'same')(net)
#net = CircleOneLayer(64)(net)
#net = KleinOneLayer(64)(net)
net = MaxPooling2D(2,2)(net)
net = Flatten()(net)
net = Dense(512, activation='relu')(net)
net = Dense(10)(net)
output = net

model1 = models.Model(img_input, output)
model1.summary()

model1.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-5),
               loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
               metrics=['accuracy'])

history1 = model1.fit(train_images_noise, train_labels, batch_size=100, epochs=5,
                      validation_data=(test_images, test_labels))
```

Program of KOL

```python
kernel_shape = (self.kernel_size[0], self.kernel_size[1], input_shape[-1], self.filters)

def Q(t):
    return 2*t**2-1

def F_K(theta_1, theta_2, x, y):
    return math.sin(theta_2)*(math.cos(theta_1)*x+math.sin(theta_1)*y)+math.cos(theta_2)*Q(math.cos(theta_1)*x+math.sin(theta_1)

def D(a_1,a_2,b_1,b_2):
    return np.power(dblquad(lambda y,x:(F_K(a_1, a_2, x, y)-F_K(b_1, b_2, x, y))**2,#函数
              -1,#x下界0
              1,#x上界pi
              lambda x:-1,#y下界x^2
              lambda x:1), 1/2)[0]#y上界2*x

temp_3 = tf.zeros(shape=kernel_shape, dtype=tf.float32).numpy()
for i in range(input_shape[-1]):
    for j in range(self.filters):
        if D(i%8 * math.pi/int(math.sqrt(input_shape[-1])), i//int(math.sqrt(input_shape[-1])) *2* math.pi/int(math.sqrt(input_s
            temp_3[:,:,i,j] = tf.ones(shape=(self.kernel_size[0], self.kernel_size[1]), dtype=tf.float32).numpy()
self.mask = tf.constant(temp_3)
self.kernel = self.add_weight(name='kernel', shape=kernel_shape, initializer='glorot_uniform',trainable=True)
self.bias = self.add_weight(name='bias', shape=(self.filters,), initializer='zeros', trainable=True)

def call(self, inputs):
```

A Result of TCNN on MNIST



Thanks for your listening!