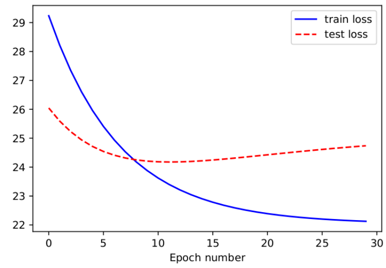
University Echahid Hama Lakhdar, El-oued
Institute of Exact Sciences
Department of Computer Science
2^{ed} *Master* : Artificial Intelligence and Data Science
Semester: 3.2024

Deep Learning Test

---

1 MCQ

Which of the following statements is a plausible explanation for what could be happening?



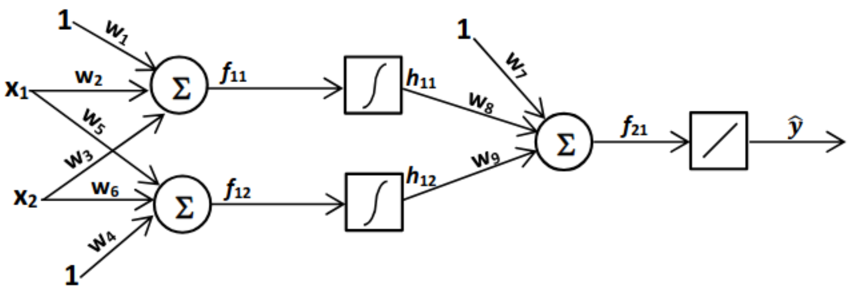
- The model is underfitting the training data.
- The model is overfitting the training data.
- The model generalizes well to unseen examples.
- None of the above.

A neural network is overfitting its training data. What strategies could mitigate this?

- Increase the dropout probability. Decrease the amount of training data.
- Increase the number of hidden units.
- All the above.

2 Exercise 1:Forward and Backward Pass

Let the multilayer neural network described by the following architecture:



1. Give the mathematical formulas that determine the intermediate outputs f_{11}, f_{12}, h_{11}, h_{12}, f_{21}, as well as the final output $\hat{y}$.
2. Let the error function be:
$$E(\mathbf{w}) = (y - \hat{y})^2$$
By applying the backpropagation algorithm, find the expressions for the parameter updates Δw_j, for $j = 1, \ldots, 9$.

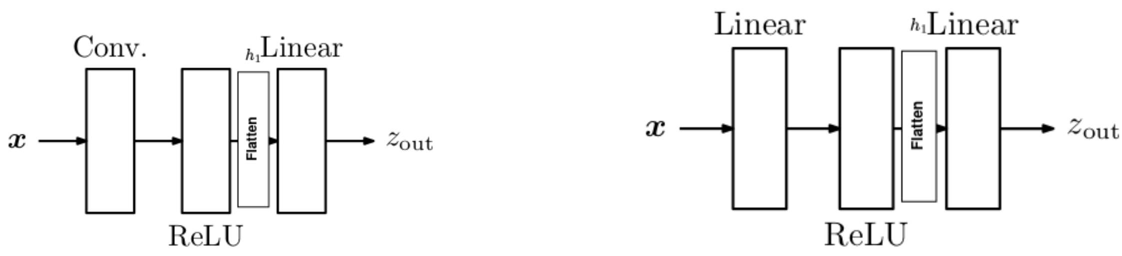
3 Problem 1: Convolutional Neural Networks (25 points)

Consider the two networks depicted in Fig. 1. In the network of Fig. 1a the first block corresponds to a convolutional layer with a single 2×2 filter, no padding and stride $s = 1$. In the network of Fig. 1b the first block corresponds to a hidden layer with 4 units and ReLU activation.

In both networks we denote by z_1 the input to the ReLU, by h_1 the input to the rightmost linear layer, which in both cases comprises a single unit. We denote by z_{out} the scalar output, such that

$$\sigma(z_{\text{out}}) = P[y = +1 \mid x],$$

where σ is the sigmoid function.

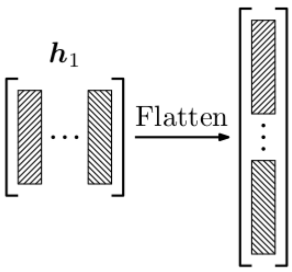


(a) Network with a convolutional layer (2x2 filter, no padding, stride 1)

(b) Network with a fully connected layer (4 ReLU units)

Figure 1: Two network architectures to process the input x.

Note: Assume that, before the linear layer, h_1 is flattened into a single column vector by stacking all columns of h_1 together, as in the following diagram:



Suppose that both networks expect as input a 3×3 matrix:

1. Write the code of two architectures.
2. Calculate the output shape and the parameter number of each layer.

2. Suppose that, after training, the parameters of the convolutional network in Fig. 1a are

$$K_1 = \begin{bmatrix} -1 & -2 \\ 1 & 2 \end{bmatrix}, \quad b_1 = 0 \qquad w_{\text{out}} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}, \quad b_{\text{out}} = -5,$$

where K_1 and b_1 are the parameters of the convolutional layer, and w_{out} and b_{out} the parameters of the linear layer. Compute the output z_{out} of the network for the input

$$x = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}.$$

4 Solution

MCQ Solutions

1. **Which of the following statements is a plausible explanation for what could be happening?**

- The model is underfitting the training data.
- **The model is overfitting the training data.** **(Correct)**
- The model generalizes well to unseen examples.
- None of the above.

2. **A neural network is overfitting its training data. What strategies could mitigate this?**
   - **Increase the dropout probability.** **(Correct)**
   - Decrease the amount of training data.
   - Increase the number of hidden units.
   - All the above.

Explanation

- **Q1:** Overfitting occurs when the model performs well on training data but poorly on unseen data.
- **Q2:** Increasing dropout is a regularization technique that helps reduce overfitting by preventing co-adaptation of neurons. The other strategies either worsen overfitting or don't directly address it.

5 Exersice 2

Solution 1

Étape 1 : Calculs dans la couche cachée

$$\begin{aligned} f_{11} &= w_1 \cdot 1 + w_2 \cdot x_1 + w_3 \cdot x_2 \\ h_{11} &= \sigma(f_{11}) \\ f_{12} &= w_4 \cdot 1 + w_5 \cdot x_1 + w_6 \cdot x_2 \\ h_{12} &= \sigma(f_{12}) \end{aligned}$$

Étape 2 : Calcul dans la couche de sortie

$$\begin{aligned} f_{21} &= w_7 \cdot 1 + w_8 \cdot h_{11} + w_9 \cdot h_{12} \\ \hat{y} &= \sigma(f_{21}) \end{aligned}$$

Solution 2

Fonction d'erreur :

$$E(\mathbf{w}) = (y - \hat{y})^2 \quad \text{où } \hat{y} = f_{21}$$

Étape 1 : Calcul du gradient à la sortie (couche de sortie)

$$\delta_2 = \frac{\partial E}{\partial f_{21}} = -2(y - \hat{y})$$

Mises à jour des poids de la couche de sortie :

$$\begin{aligned} \Delta w_7 &= -\eta \cdot \delta_2 \cdot 1 \\ \Delta w_8 &= -\eta \cdot \delta_2 \cdot h_{11} \\ \Delta w_9 &= -\eta \cdot \delta_2 \cdot h_{12} \end{aligned}$$

Étape 2 : Propagation de l'erreur vers la couche cachée

$$\begin{aligned} \delta_{11} &= \delta_2 \cdot w_8 \cdot \sigma'(f_{11}) \\ \delta_{12} &= \delta_2 \cdot w_9 \cdot \sigma'(f_{12}) \end{aligned}$$

Mises à jour des poids d'entrée :

$$\begin{aligned}
\Delta w_1 &= -\eta \cdot \delta_{11} \cdot 1 \\
\Delta w_2 &= -\eta \cdot \delta_{11} \cdot x_1 \\
\Delta w_3 &= -\eta \cdot \delta_{11} \cdot x_2 \\
\Delta w_4 &= -\eta \cdot \delta_{12} \cdot 1 \\
\Delta w_5 &= -\eta \cdot \delta_{12} \cdot x_1 \\
\Delta w_6 &= -\eta \cdot \delta_{12} \cdot x_2
\end{aligned}$$

6 Problem

1. Codes:

Keras Implementation

Convolutional Network (Fig. 1a):

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, ReLU, Flatten, Dense

# Convolutional model: Conv2D (2x2 kernel), ReLU, Flatten, Dense
conv_model = Sequential([
    Conv2D(filters=1, kernel_size=(2, 2), strides=(1, 1), padding='valid',
           input_shape=(3, 3, 1), use_bias=True, name="conv"),
    ReLU(),
    Flatten(),
    Dense(1, name="fc")
])

conv_model.summary()
```

Fully Connected Network (Fig. 1b):

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, ReLU, Flatten, InputLayer

# Fully connected model: Flatten → Dense (4 units + ReLU) → Dense (1 unit)
fc_model = Sequential([
    InputLayer(input_shape=(3, 3, 1)),
    Flatten(),
    Dense(4),
    ReLU(),
    Dense(1)
])

fc_model.summary()
```

2. **Calculate the output shape and the parameter number of each layer**

   - **Network in Fig. 1a (Convolutional Network):**
     - *Input:* 3×3 matrix.
     - *Conv2D layer:* One 2×2 filter, stride 1, no padding.
       * Output shape: $(3-2+1) \times (3-2+1) = 2 \times 2$
       * Number of parameters: $2 \times 2 = 4$ (weights) $+1$ (bias) $= 5$
     - *ReLU layer:* Output shape unchanged: 2×2
     - *Flatten layer:* Converts 2×2 into 4×1 vector
     - *Dense layer:* Input dimension 4, output dimension 1
       * Number of parameters: 4 (weights) $+1$ (bias) $= 5$
     - **Total parameters:** 5 (Conv2D) $+5$ (Dense) $= \boxed{10}$
   - **Network in Fig. 1b (Fully Connected Network):**

- *Input:* 3×3 matrix flattened to a 9×1 vector
- *First Dense layer:* Input 9, output 4
  - Number of parameters: $9 \times 4 = 36$ (weights) $+4$ (biases) $= 40$
- *ReLU activation:* Output shape unchanged: 4
- *Second Dense layer:* Input 4, output 1
  - Number of parameters: 4 (weights) $+1$ (bias) $= 5$
- **Total parameters:** 40 (first Dense) $+5$ (second Dense) $= \boxed{45}$

3. **Summary of the two models:**

Model	**Layer**	**Output Shape**	**# Parameters**
3*Convolutional Network	Conv2D (1 filter, 2x2)	(2, 2, 1)	5
	Flatten	(4,)	0
	Dense (1 unit)	(1,)	5
	Total	—	**10**
3*Fully Connected Network	Dense (4 units)	(4,)	40
	ReLU	(4,)	0
	Dense (1 unit)	(1,)	5
	Total	—	**45**

Table 1: Model summaries: output shapes and parameter counts

4. **How to Compute the Number of Parameters in Each Layer**

- **1. Convolutional Layer**
  For a convolutional layer with:
  - Filter size: $k \times k$
  - Input channels: C_{in}
  - Output channels (filters): C_{out}
  - One bias per output channel

  The total number of parameters is:

  $$\#\text{parameters} = (k \cdot k \cdot C_{\text{in}}) \cdot C_{\text{out}} + C_{\text{out}}$$

  Example: For a 2×2 filter, $C_{\text{in}} = 1$, $C_{\text{out}} = 1$:

  $$\#\text{parameters} = (2 \cdot 2 \cdot 1) \cdot 1 + 1 = 4 + 1 = 5$$

- **2. Fully Connected (Dense) Layer**
  For a dense layer with:
  - Number of input units: n_{in}
  - Number of output units: n_{out}
  - One bias per output unit

  The total number of parameters is:

  $$\#\text{parameters} = n_{\text{in}} \cdot n_{\text{out}} + n_{\text{out}}$$

  Example: For $n_{\text{in}} = 4$, $n_{\text{out}} = 1$:

  $$\#\text{parameters} = 4 \cdot 1 + 1 = 5$$

- **3. Flatten Layer**
  The flatten layer only reshapes the tensor and has:

  $$\#\text{parameters} = 0$$

- **Total for the Convolutional Model:**

- Conv2D: 5 parameters
- Flatten: 0 parameters
- Dense: 5 parameters

$$\text{Total parameters} = 5 + 0 + 5 = \boxed{10}$$

5. **Parameter Calculation for the Architecture in Fig. 1b (Fully Connected Network)**
The network consists of:
   - An input layer that takes a $3 \times 3 = 9$-dimensional input vector (the 3×3 matrix is flattened),
   - A fully connected hidden layer with 4 ReLU units,
   - A final output layer with a single unit.

   (a) **First Dense Layer (Input to Hidden Layer)**:
   - Number of input units: $n_{\text{in}} = 9$
   - Number of output units: $n_{\text{out}} = 4$
   - Parameters: $9 \cdot 4$ weights + 4 biases

   $$\Rightarrow \#\text{parameters} = 9 \cdot 4 + 4 = 36 + 4 = 40$$

   (b) **ReLU Activation**:
   - No parameters

   $$\Rightarrow \#\text{parameters} = 0$$

   (c) **Second Dense Layer (Hidden to Output)**:
   - Input: 4 units, Output: 1 unit
   - Parameters: $4 \cdot 1$ weights + 1 bias

   $$\Rightarrow \#\text{parameters} = 4 + 1 = 5$$

   Total number of parameters:

   $$\boxed{40 + 0 + 5 = 45}$$

6. **Compute the output z_{out} of the network for the input**
   - **Input:**
   $$x = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$
   - **Convolution filter:**
   $$K_1 = \begin{bmatrix} -1 & -2 \\ 1 & 2 \end{bmatrix}, \quad b_1 = 0$$
   - **Convolution output (before ReLU):**
   The 2×2 output from valid convolution (stride 1) is computed as follows:
   $$z_1 = \begin{bmatrix} (-1)(1) + (-2)(0) + (1)(0) + (2)(1) & (-1)(0) + (-2)(1) + (1)(1) + (2)(0) \\ (-1)(0) + (-2)(1) + (1)(1) + (2)(0) & (-1)(1) + (-2)(0) + (1)(0) + (2)(1) \end{bmatrix}$$
   $$z_1 = \begin{bmatrix} -1 + 0 + 0 + 2 & 0 - 2 + 1 + 0 \\ 0 - 2 + 1 + 0 & -1 + 0 + 0 + 2 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$
   - **After ReLU:**
   $$h_1 = \text{ReLU}(z_1) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$
   - **Flattened vector:** Flatten by stacking columns:
   $$\text{vec}(h_1) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$
   - **Linear output:**
   $$w_{\text{out}} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}, \quad b_{\text{out}} = -5$$
   $$z_{\text{out}} = w_{\text{out}}^\top \cdot \text{vec}(h_1) + b_{\text{out}} = (1)(1) + (2)(0) + (3)(0) + (4)(1) - 5 = 0$$
   - **Final result:**
   $$\boxed{z_{\text{out}} = 0}$$

University Echahid Hama Lakhdar, El-oued
Institute of Exact Sciences
Department of Computer Science
2^{ed} *Master* : Artificial Intelligence and Data Science
Semester: 3.2024

Deep Learning Test

Section 1: Multiple Choice Questions (0.75×7 points)

1. Which activation function is most likely to cause "dead neurons" in training?
   a) ReLU b) Sigmoid c) Tanh d) Leaky ReLU
2. The vanishing gradient problem is most severe in:
   a) CNNs with skip connections b) Transformers with self-attention
   c) Deep FFNs with sigmoid activations d) RNNs trained on long sequences
3. Output size of conv layer with input $32 \times 32 \times 3$, 10 filters, 5×5 kernel, stride 2, valid padding:
   a) $14 \times 14 \times 10$ b) $15 \times 15 \times 10$
   c) $28 \times 28 \times 10$ d) $16 \times 16 \times 10$
4. Appropriate loss for multi-class classification:
   a) MSE b) Binary CE c) Categorical CE d) Hinge Loss
5. Reduces overfitting in CNNs:
   a) More depth b) Dropout c) Remove BN d) Larger LR
6. Gradient for FC layer weights depends on:
   a) Input + upstream b) Upstream only c) Input only d) Output weights
7. High training/low validation accuracy indicates:
   a) Underfitting b) Overfitting c) High bias d) Optimal

Section 2: Forward and Backward Pass (8.75)

Consider a feedforward neural network with the following architecture:

- Input layer: 3 neurons.
- First hidden layer: 4 neurons with ReLU activation.
- Second hidden layer: 2 neurons with sigmoid activation.
- Output layer: 1 neuron with linear activation (i.e., identity).

You are given the following:

- Input vector:

$$x = \begin{bmatrix} 1 \\ -1 \\ 2 \end{bmatrix}$$

- Weight matrix and bias vector for the first layer:

$$W^{[1]} = \begin{bmatrix} 0.2 & -0.3 & 0.5 \\ -0.4 & 0.1 & 0.6 \\ 0.7 & 0.8 & -0.2 \\ 0.3 & -0.5 & 0.4 \end{bmatrix}, \quad b^{[1]} = \begin{bmatrix} 0.1 \\ -0.2 \\ 0.05 \\ 0 \end{bmatrix}$$

- Weight matrix and bias vector for the second layer:

$$W^{[2]} = \begin{bmatrix} 0.6 & -0.1 & 0.3 & 0.7 \\ -0.5 & 0.2 & 0.4 & -0.6 \end{bmatrix}, \quad b^{[2]} = \begin{bmatrix} 0.1 \\ 0.3 \end{bmatrix}$$

- Weight and bias for the output layer:

$$W^{[3]} = \begin{bmatrix} 0.2 & -0.4 \end{bmatrix}, \quad b^{[3]} = 0.05$$

- The ground truth label (target output) is:

$$y = 0.6$$

- Use Mean Squared Error (MSE) as the loss function:

$$L = \frac{1}{2}(\hat{y} - y)^2$$

1. Compute the output $\hat{y}$ of the network using the given input vector and weights.
2. Compute the gradients of the loss with respect to all weights and biases in the network using backpropagation.

Section 3: CNN Problem (06 points)

Architecture:

- Input: $64 \times 64 \times 3$ RGB
- Conv1: 32 filters, 3×3, stride 1, same padding, ReLU
- MaxPool: 2×2, stride 2
- Conv2: 64 filters, 5×5, stride 2, valid padding, ReLU
- Global Avg Pool
- Output: 10 neurons, softmax

Questions:

1. Write the code of the architecture
2. Output dimensions after each layer
3. Total trainable parameters
4. Explain "same" padding in Conv1
5. Propose 2 overfitting solutions

1 Solution

1. Which activation function is most likely to cause "dead neurons" in training?
   Answer: a) ReLU
   Explanation: ReLU can output zero for all negative inputs, and once a neuron gets stuck in this state, it may never recover.
2. The vanishing gradient problem is most severe in:
   Answer: d) RNNs trained on long sequences
   Explanation: Long-term dependencies in RNNs cause gradients to shrink exponentially, making learning difficult.
3. Output size of conv layer with input $32 \times 32 \times 3$, 10 filters, 5×5 kernel, stride 2, valid padding:
   Answer: a) $14 \times 14 \times 10$
   Explanation: Output height/width $= \left\lfloor \frac{32-5}{2} + 1 \right\rfloor = 14$*; depth = number of filters = 10.*
4. Appropriate loss for multi-class classification:
   Answer: c) Categorical CE
   Explanation: Categorical cross-entropy is suited for multi-class problems with one-hot encoded labels.
5. Reduces overfitting in CNNs:
   Answer: b) Dropout
   Explanation: Dropout regularizes the model by randomly deactivating neurons during training.
6. Gradient for FC layer weights depends on:
   Answer: a) Input + upstream
   Explanation: The gradient of weights is computed as the product of upstream gradients and the input to the layer.
7. High training/low validation accuracy indicates:
   Answer: b) Overfitting
   Explanation: The model memorizes training data but generalizes poorly to unseen validation data.

Forward Pass and Password

Given input:

$$x = \begin{bmatrix} 1 \\ -1 \\ 2 \end{bmatrix}$$

First layer:

$$z^{[1]} = W^{[1]}x + b^{[1]} = \begin{bmatrix} 0.2 & -0.3 & 0.5 \\ -0.4 & 0.1 & 0.6 \\ 0.7 & 0.8 & -0.2 \\ 0.3 & -0.5 & 0.4 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 2 \end{bmatrix} + \begin{bmatrix} 0.1 \\ -0.2 \\ 0.05 \\ 0 \end{bmatrix}$$

Compute:

$$W^{[1]}x = \begin{bmatrix} 0.2(1) + (-0.3)(-1) + 0.5(2) = 1.5 \\ -0.4(1) + 0.1(-1) + 0.6(2) = 0.7 \\ 0.7(1) + 0.8(-1) + (-0.2)(2) = -0.5 \\ 0.3(1) + (-0.5)(-1) + 0.4(2) = 1.6 \end{bmatrix}$$

Add bias:

$$z^{[1]} = \begin{bmatrix} 1.5 + 0.1 = 1.6 \\ 0.7 - 0.2 = 0.5 \\ -0.5 + 0.05 = -0.45 \\ 1.6 + 0 = 1.6 \end{bmatrix}$$

Apply ReLU activation:

$$a^{[1]} = \text{ReLU}(z^{[1]}) = \begin{bmatrix} 1.6 \\ 0.5 \\ 0 \\ 1.6 \end{bmatrix}$$

Second layer:

$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]} = \begin{bmatrix} 0.6 & -0.1 & 0.3 & 0.7 \\ -0.5 & 0.2 & 0.4 & -0.6 \end{bmatrix} \begin{bmatrix} 1.6 \\ 0.5 \\ 0 \\ 1.6 \end{bmatrix} + \begin{bmatrix} 0.1 \\ 0.3 \end{bmatrix}$$

Compute:

$$W^{[2]}a^{[1]} = \begin{bmatrix} 0.6(1.6) + (-0.1)(0.5) + 0(0.3) + 0.7(1.6) = 0.96 - 0.05 + 0 + 1.12 = 2.03 \\ -0.5(1.6) + 0.2(0.5) + 0(0.4) + (-0.6)(1.6) = -0.8 + 0.1 + 0 - 0.96 = -1.66 \end{bmatrix}$$

Add bias:

$$z^{[2]} = \begin{bmatrix} 2.03 + 0.1 = 2.13 \\ -1.66 + 0.3 = -1.36 \end{bmatrix}$$

Apply sigmoid:

$$a^{[2]} = \sigma(z^{[2]}) = \begin{bmatrix} \sigma(2.13) \\ \sigma(-1.36) \end{bmatrix} = \begin{bmatrix} \frac{1}{1+e^{-2.13}} \approx 0.894 \\ \frac{1}{1+e^{1.36}} \approx 0.204 \end{bmatrix}$$

Output layer (linear):

$$\hat{y} = W^{[3]}a^{[2]} + b^{[3]} = \begin{bmatrix} 0.2 & -0.4 \end{bmatrix} \begin{bmatrix} 0.894 \\ 0.204 \end{bmatrix} + 0.05 = 0.2(0.894) - 0.4(0.204) + 0.05$$

$$\hat{y} = 0.1788 - 0.0816 + 0.05 = 0.1472$$

Loss:

$$L = \frac{1}{2}(\hat{y} - y)^2 = \frac{1}{2}(0.1472 - 0.6)^2 \approx \frac{1}{2}(-0.4528)^2 \approx 0.1024$$

2 Problem

Architecture:

- Input: $64 \times 64 \times 3$ RGB
- Conv1: 32 filters, 3×3, stride 1, same padding, ReLU
- MaxPool: 2×2, stride 2
- Conv2: 64 filters, 5×5, stride 2, valid padding, ReLU

- Global Average Pooling
- Output: 10 neurons, softmax

Questions:

1. **Write the code of the architecture (using Keras):**

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, GlobalAveragePooling2D, Dense, Activation

model = Sequential([
    Conv2D(32, (3, 3), strides=1, padding='same', activation='relu', input_shape=(64, 64, 3)),
    MaxPooling2D(pool_size=(2, 2), strides=2),
    Conv2D(64, (5, 5), strides=2, padding='valid', activation='relu'),
    GlobalAveragePooling2D(),
    Dense(10, activation='softmax')
])
```

2. Output Dimensions

Layer	Output Dimensions
Input	$64 \times 64 \times 3$
Conv1 (ReLU)	$64 \times 64 \times 32$
MaxPool	$32 \times 32 \times 32$
Conv2 (ReLU)	$14 \times 14 \times 64$
Global Average Pool	64 (vector)
Output (Softmax)	10 (probabilities)

3. Parameter Calculations

Conv1:

$$\begin{aligned} \text{Weights} &= (3 \times 3 \times 3) \times 32 = 864 \\ \text{Biases} &= 32 \\ \text{Total} &= 864 + 32 = 896 \end{aligned}$$

Conv2:

$$\begin{aligned} \text{Weights} &= (5 \times 5 \times 32) \times 64 = 51{,}200 \\ \text{Biases} &= 64 \\ \text{Total} &= 51{,}200 + 64 = 51{,}264 \end{aligned}$$

Output Layer:

$$\begin{aligned} \text{Weights} &= 64 \times 10 = 640 \\ \text{Biases} &= 10 \\ \text{Total} &= 640 + 10 = 650 \end{aligned}$$

Global Total:

$$896 + 51{,}264 + 650 = \boxed{52{,}810}$$

4. "Same" Padding Explanation

For `Conv1` with:

- Kernel size: 3×3
- Stride: 1
- Input size: 64×64

Padding required to maintain spatial dimensions:

$$P = \left\lfloor \frac{\text{Kernel Size}}{2} \right\rfloor = \left\lfloor \frac{3}{2} \right\rfloor = 1$$

This adds 1 pixel of zero-padding *on all sides*, ensuring:

$$\text{Output Size} = \left\lfloor \frac{64 - 3 + 2 \times 1}{1} \right\rfloor + 1 = 64$$

5. Overfitting Solutions

(a) **Dropout:** Add after Global Average Pooling:

```
layers.Dropout(0.5)
```

(b) **Data Augmentation:** Use preprocessing layers:

```
data_augmentation = tf.keras.Sequential([
    layers.RandomFlip("horizontal"),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.2)
])
```

Final Summary

- Code: See Section 1
- Dimensions: $64^3 \to 64^{32} \to 32^{32} \to 14^{64} \to 64 \to 10$
- Parameters: $\boxed{52{,}810}$
- Same Padding: $\boxed{\text{Maintains input dimensions with 1-pixel padding}}$
- Overfitting Fixes: $\boxed{\text{Dropout \& Data Augmentation}}$

2. **Explain "same" padding in Conv1**:
   "Same" padding means padding the input so that the output has the same spatial dimensions as the input when the stride is 1. For a 3×3 kernel, one pixel of zero-padding is added to all sides. It preserves the input width and height.
3. **Propose 2 overfitting solutions**:
   - Add **Dropout layers** to randomly deactivate neurons during training.
   - Use **Data Augmentation** (rotation, flipping, cropping) to synthetically increase dataset size and diversity.