1.

What does *translation invariance* mean in the context of designing neural network architectures for processing images?

- [x] Network should respond similarly to the same image patch, regardless of where the patch appears in the image
- [ ] Network should network should focus on local regions, without regard for the contents of the image in distant regions
- [ ] Network should exploit some of the known structure in natural images
- [ ] Network should be easy to train on image data

2.

What does *locality* mean in the context of designing neural network architectures for processing images?

- [ ] Network should respond similarly to the same image patch, regardless of where the patch appears in the image
- [x] Network should focus on local regions, without regard for the contents of the image in distant regions
- [ ] Network should exploit some of the known structure in natural images
- [ ] Network should be easy to train on image data

3.

Which of the following statements is correct?

- [x] For the same input and output dimensions, a convolutional layer will typically have fewer parameters than a fully connected layer
- [ ] For the same input and output dimensions, a convolutional layer can express more functions than a fully connected layer
- [ ] For the same input and output dimensions, a convolutional layer will learn a better hidden representation given any data
- [ ] For the same input and output dimensions, a convolutional layer will learn a better representation on tabular data where we do not assume any structure *a priori* on how features interact

4.

If a 100 x 100 pixel grayscale image is to be converted into a hidden representation of dimension 100 using a fully connected layer, roughly how many parameters will be needed in that layer?

- [x] 10^6
- [ ] 10^4
- [ ] 10^2
- [ ] 10^8

parameter num = input dim × output dim

$= 10^4 \times 10^2 = 10^6$

5.

Complete the following analogy.

Dense layer : multilayer perceptrons :: convolutional layer : ?

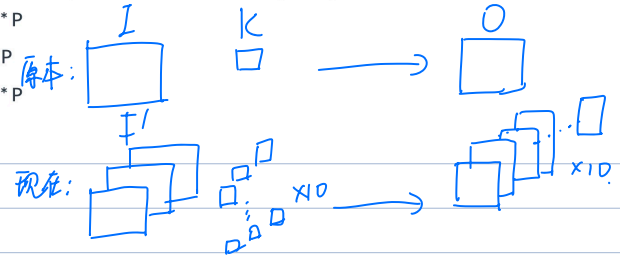
- [x] convolutional neural networks
- [ ] recurrent neural networks
- [ ] artificial neural networks
- [ ] deep neural networks

6.

Suppose we design a convolution kernel K with P parameters/weights for use with a 1024 x 1024 grayscale image. Then the image shape is changed to 1024 x 1024 x 3 to accommodate 3 color channels. Keeping the spatial dimensions of K the same, we design a new kernel K' to operate on color images and to output 10 channels. Let P' be the number of parameters/weights in K'. (Focus on weights for this question, ignore biases).

Which of the following is a correct statement regarding the relationship between P and P'?

- [x] P' = 30 * P
- [ ] P' = 3 * P
- [ ] P' = 10 * P
- [ ] P' = P



7.

Suppose f and g are functions over the integers. Which of the following expression correctly computes their convolution evaluated at an integer i?

- [x] Σ_a f(i-a) g(a)
- [ ] Σ_a f(a)g(i+a)
- [ ] Σ_a f(a)g(a-i)
- [ ] None of these expressions computes the convolution correctly

sum 在 dummy variable a 上

f(i-a), g(a)

8.

What is the price we pay for massively reducing the number of parameters in a convolutional layer (compared to a fully connected layer)?

- [x] If data does not follows the assumptions of translation invariance and locality, our models might struggle even to fit our training data.
- [ ] Training a convolutional layer is harder than training a fully connected layer
- [ ] We can no longer use regularization to control overfitting in a convolutional layer
- [ ] We can no longer use non-linear activations such as the ReLU (rectified linear unit)

9.

Fix a value of x and consider the function f(w) = $(w-x)^2$. Clearly the minimizer of f(w) is x and the minimum value of f is zero. What will be the minimizer of the regularized function g(w) = $(w-x)^2$ + lambda * w^2 ?

- x
- x + lambda * w^2
- x + lambda * x
- **x / (1 + lambda)**

$g(w) = (w-x)^2 + \lambda w^2$

$= w^2 - 2wx + x^2 + \lambda w^2$

$= (1+\lambda) w^2 - 2wx + x^2$

$w = \frac{2x}{2(1+\lambda)}$ 时 $g(w)$ 最小

$= \frac{x}{1+\lambda}$

L_1, L_2 Regularization 就是：

在 activation function 被加入前，一个 layer 就是一个 regression.

而 L_1 / L_2 regression 就是在这个 regression 后加入一个 penalty term，然后再应用 activation function

L_1 regularization: $L(W) = \sum_{i=1}^{n} (Y_i - \sum_{j=1}^{p} W_j X_{ij}) + \lambda \sum_{j=1}^{p} |W_j|$

(aka lasso regression)　(sparsity ↑)　一些 weights 会变为 0，忽略一些 features.

λ ~ hyperparameter

10.

Suppose a reference model had a layer like this:

```
layers.Dense(16, activation="relu")
```

What happens if you change this layer to the following to get a new model:

```
layers.Dense(16, kernel_regularizer=regularizers.l2(0.0), activation="relu")
```

等于没加

- You get an error
- **The new model is the same as the reference model**
- The new model underfits more compared to the reference model
- The new model overfits more compared to the reference model

使 W 分布更均匀（防止一个 weight 过大）

L_2 (regularization): $L(W) = \sum_{i=1}^{n} (Y_i - \sum_{j=1}^{p} W_j X_{ij}) + \lambda \sum_{j=1}^{p} W_j^2$

(aka ridge regression)

λ ~ hyperparameter