**1. Why use convolution instead of fully connected layers for images?**

* Convolutions exploit **spatial locality** (nearby pixels are correlated).
* They **share weights**, reducing parameters drastically.
* They achieve **translation invariance** (detect features regardless of location).

**2. What is the difference between convolution and cross-correlation?**

* In math, convolution flips the kernel before sliding.
* In deep learning libraries (TensorFlow, PyTorch), we actually use **cross-correlation** (no flip).
* In practice, the flip doesn’t matter because kernels are learned.

**3. What is receptive field in CNNs?**

* The region of the input image that affects one unit in a feature map.
* Increases with more layers or larger filters.

**4. Why use small filters (3×3) instead of large ones (7×7, 11×11)?**

* Multiple 3×3 layers approximate a larger filter (e.g., 3 layers of 3×3 ≈ 7×7).
* Fewer parameters → less overfitting.
* More non-linearity (activation between layers).
* Used in VGGNet.

**5. What is padding in CNNs and why is it important?**

* Padding = adding zeros around the image.
* **Same padding** keeps output size equal to input size.
* Prevents shrinking feature maps too fast.
* Preserves edge information.

**6. What is stride in CNN?**

* Stride = step size when sliding filter.
* Higher stride → reduces spatial dimension (downsampling).
* Example: stride 2 halves the feature map size.

**7. What is the difference between max pooling and average pooling?**

* **Max pooling** keeps strongest activation (detects presence of features).
* **Average pooling** takes mean (smooths features).
* Max pooling is more common in CNNs.

**8. Why batch normalization is useful in CNNs?**

* Normalizes activations across a batch.
* Reduces **internal covariate shift**.
* Speeds up convergence and improves stability.
* Acts as a regularizer.

**9. What is the difference between CNNs and Vision Transformers (ViTs)?**

* CNNs use **local inductive bias** (convolutions capture local features).
* ViTs use **self-attention** (global context from the start).
* CNNs are data-efficient for small datasets, ViTs excel on large datasets.

**10. What are dilated convolutions and where are they used?**

* Convolutions with “gaps” (dilation rate > 1).
* Increase receptive field without increasing parameters.
* Used in **semantic segmentation** (e.g., DeepLab).

**1. What are Depthwise Separable Convolutions? Why are they efficient?**

* Normal Conv: applies filter across **all channels** → very expensive.
* **Depthwise Convolution**: applies a filter per channel (spatial filtering).
* **Pointwise Convolution (1×1)**: mixes channels.
* **Efficiency**: Reduces parameters & FLOPs drastically.
* Example: **MobileNet** uses this for lightweight models.

**2. What are Transposed Convolutions (a.k.a. Deconvolutions)?**

* Used for **upsampling** (e.g., generating larger feature maps).
* Opposite of convolution: maps **low-resolution features → high-resolution**.
* Common in **image segmentation (U-Net), GANs**.
* Challenge: can cause **checkerboard artifacts** if not carefully designed.

**3. What problem do Residual Connections (ResNet) solve?**

* In very deep CNNs → gradients **vanish or explode**.
* ResNet introduces **skip connections**:

y=F(x)+xy

* Helps gradient flow directly → allows training of 100+ layers.
* Key innovation in deep learning after 2015.

**4. What is a Bottleneck Layer in CNNs?**

* Used in **ResNet, DenseNet**.
* Instead of large 3×3 conv directly, use:
  + 1×1 conv (reduce channels) → 3×3 conv → 1×1 conv (restore channels).
* Cuts parameters while keeping accuracy.

**Why don’t we use even-sized filters (2×2, 4×4, …) in CNNs?**

**1. Symmetry & Center Pixel Problem**

* With **odd-sized filters** (3×3, 5×5, 7×7):
  + There is a **clear center pixel**.
  + Example: 3×3 → center = (2,2).
* With **even-sized filters** (2×2, 4×4):
  + No exact center.
  + Filter is “off-center” → leads to **asymmetric receptive field**.
  + Causes **shifts in feature maps** layer by layer.

**2. Padding Issues**

* CNNs often use "same" padding (zero-padding to preserve size).
* Odd filters: symmetric padding is easy → e.g., 3×3 adds 1 pixel on each side.
* Even filters: asymmetric padding required → e.g., 2×2 requires padding **more on one side than the other**.
* This causes **spatial bias** in outputs.

**3. Loss of Translational Invariance**

* One of CNNs’ strengths: **translation invariance**.
* Even filters → asymmetric padding → breaks this invariance.
* The network may behave differently if the same object shifts by 1 pixel.

**4. Historical & Practical Reasons**

* Early CNNs (LeNet, AlexNet, VGG) standardized on **3×3 filters** → became a convention.
* 3×3 is also a sweet spot:
  + Small enough to stack for large receptive fields.
  + Large enough to capture meaningful spatial patterns.
* Example: 2 stacked 3×3 convs ≈ 5×5 conv but with fewer parameters & more non-linearity.

**✅ Summary**

We avoid **even-sized filters** in CNNs because:

1. No clear center pixel → breaks symmetry.
2. Asymmetric padding required → spatial bias.
3. Translation invariance is lost.
4. Odd filters (esp. 3×3) are more efficient and standard.

⚡ Bonus: **Are even filters ever used?**

* Rare, but yes — in some special cases:
  + **Subsampling layers (2×2 pooling)**.
  + **Shift-invariant CNNs** may carefully handle padding.
  + Some architectures experiment with 2×2 conv for efficiency, but it’s unstable.

Would you like me to also prepare