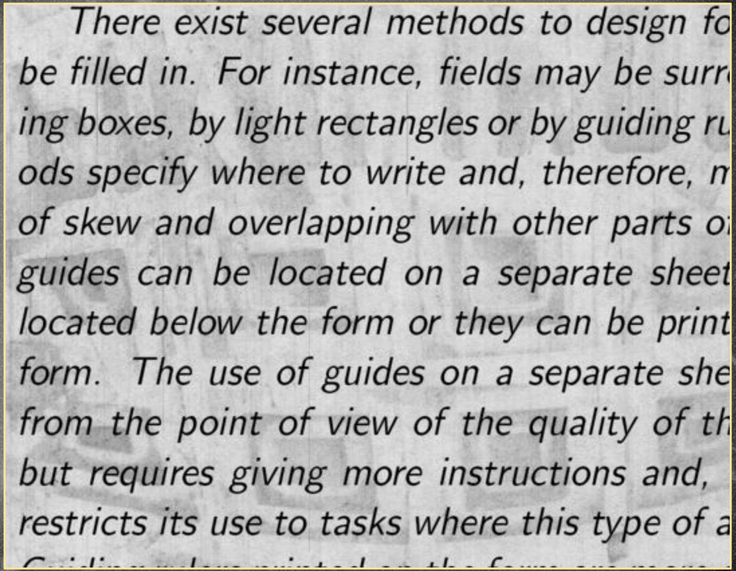
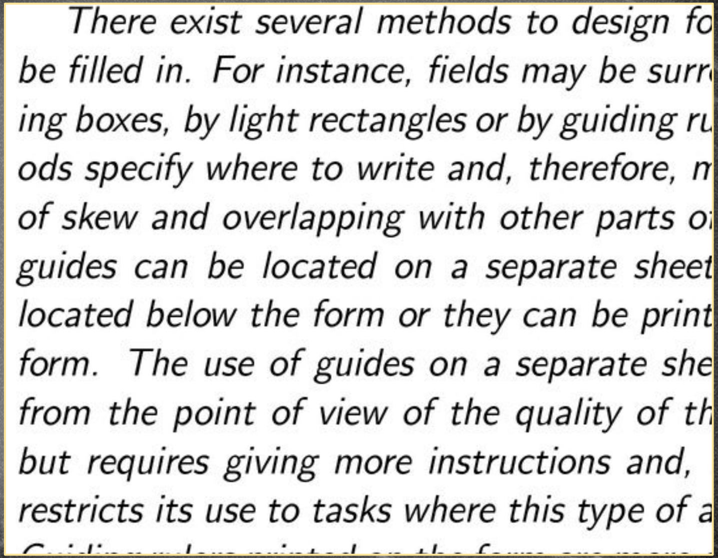
IMAGE DENOISING


There exist several methods to design fo
be filled in. For instance, fields may be surr
ing boxes, by light rectangles or by guiding ru
ods specify where to write and, therefore, m
of skew and overlapping with other parts o
guides can be located on a separate sheet
located below the form or they can be print
form. The use of guides on a separate she
from the point of view of the quality of th
but requires giving more instructions and,
restricts its use to tasks where this type of a


There exist several methods to design fo
be filled in. For instance, fields may be surr
ing boxes, by light rectangles or by guiding ru
ods specify where to write and, therefore, m
of skew and overlapping with other parts o
guides can be located on a separate sheet
located below the form or they can be print
form. The use of guides on a separate she
from the point of view of the quality of th
but requires giving more instructions and,
restricts its use to tasks where this type of a

ENCODER

python

```python
# Encoder
x = Conv2D(32, (3,3), activation='relu', padding='same', name='Conv1')(input_img)
x = MaxPooling2D((2,2), padding='same', name='pool1')(x)
x = Conv2D(64, (3,3), activation='relu', padding='same', name='Conv2')(x)
x = MaxPooling2D((2,2), padding='same', name='pool2')(x)
```

DECODER

python

```python
# Decoder
x = Conv2D(64, (3,3), activation='relu', padding='same', name='Conv3')(x)
x = UpSampling2D((2,2), name='upsample1')(x)
x = Conv2D(32, (3,3), activation='relu', padding='same', name='Conv4')(x)
x = UpSampling2D((2,2), name='upsample2')(x)
x = Conv2D(1, (3,3), activation='sigmoid', padding='same', name='Conv5')(x)
```

The **encoder** part of the network compresses the input image into a lower-dimensional representation (latent space). This is crucial for denoising because:

- It extracts **important features** from the noisy image while discarding irrelevant noise.
- The pooling layers in the encoder reduce the spatial dimensions, which helps in focusing on the **global structure** of the image rather than local noise.

Example:

- Input: Noisy image (420x540x1)
- Encoder Output: Compressed representation (105x135x64)

The **decoder** part reconstructs the clean image from the compressed representation. It does this by:

- Upsampling the compressed features back to the original image size.
- Using learned patterns to **reconstruct the clean image** while suppressing noise.

Example:

- Decoder Input: Compressed representation (105x135x64)
- Decoder Output: Denoised image (420x540x1)

Noise Removal Mechanism

The encoder-decoder architecture inherently learns to separate noise from the image because:

- Noise is typically **high-frequency and random**, while the actual image content is **structured and low-frequency**.
- The encoder's pooling layers **suppress high-frequency noise** by downsampling.
- The decoder reconstructs the image using the **low-frequency, noise-free features** extracted by the encoder.

End-to-End Learning

The encoder-decoder architecture allows the model to learn the entire denoising process in an **end-to-end manner**:

- The model is trained on pairs of noisy and clean images.
- It learns to map noisy inputs to clean outputs directly, without requiring handcrafted features or preprocessing steps.

Why Not Use a Simple CNN?

A simple CNN without an encoder-decoder structure:

- May not effectively compress and reconstruct the image.
- Could overfit to noise instead of learning to remove it.
- Lacks the **bottleneck layer**, which is critical for forcing the network to focus on important features.

Thank You for Your Attention!

We hope this presentation provided valuable insights into the power of **Encoder-Decoder Architectures** for Image Denoising.