DeepLearning.AI

CNNs: Feature Maps and Receptive Fields

Specialized Approaches to Vision

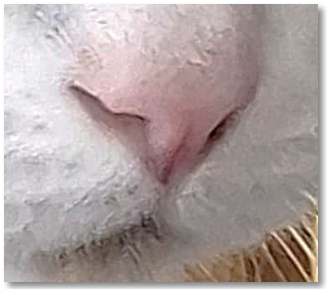
Module 2: Specialized Approaches to Vision

- **CNNs:** Feature Maps and Receptive Fields
- **Interpretability:** Saliency & Class Activation Maps
- **Diffusion Models:** Image Generation

Laurence Moroney

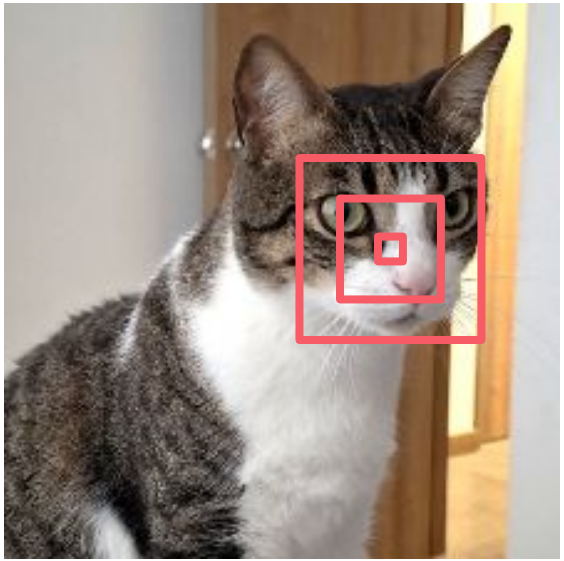
CNN Filters

-1	0	1
-2	0	2
-1	0	1



Detection Scale

-1	0	1
-2	0	2
-1	0	1

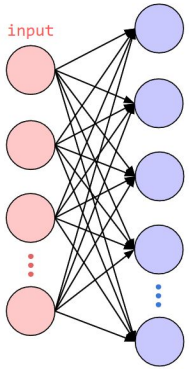


Convolutions vs. Linear Layers

224x224 image

R: 188 G: 179 B: 172	R: 192 G: 178 B: 166	R: 200 G: 180 B: 164	R: 187 G: 164 B: 149	R: 170 G: 149 B: 134
R: 208 G: 201 B: 195	R: 215 G: 197 B: 184	R: 211 G: 191 B: 176	R: 206 G: 185 B: 170	R: 181 G: 161 B: 145
R: 232 G: 228 B: 225	R: 229 G: 216 B: 207	R: 224 G: 207 B: 196	R: 216 G: 200 B: 189	R: 204 G: 192 B: 182
R: 252 G: 249 B: 250	R: 240 G: 234 B: 231	R: 229 G: 224 B: 219	R: 228 G: 223 B: 220	R: 226 G: 223 B: 222
R: 249 G: 247 B: 250	R: 232 G: 228 B: 230	R: 232 G: 230 B: 231	R: 231 G: 229 B: 230	R: 223 G: 220 B: 221

150,000+ RGB Values



64 Neurons

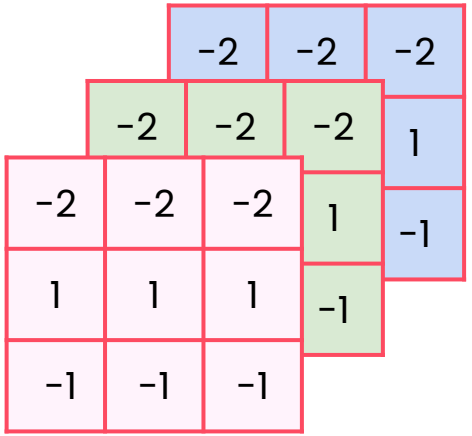
~10 Million Weights!

Convolutions vs. Linear Layers

224x224 image

R: 188 G: 179 B: 172	R: 192 G: 178 B: 166	R: 200 G: 180 B: 164	R: 187 G: 164 B: 149	R: 170 G: 149 B: 134
R: 208 G: 201 B: 195	R: 215 G: 197 B: 184	R: 211 G: 191 B: 176	R: 206 G: 185 B: 170	R: 181 G: 161 B: 145
R: 232 G: 228 B: 225	R: 229 G: 216 B: 207	R: 224 G: 207 B: 196	R: 216 G: 200 B: 189	R: 204 G: 192 B: 182
R: 252 G: 249 B: 250	R: 240 G: 234 B: 231	R: 229 G: 224 B: 219	R: 228 G: 223 B: 220	R: 226 G: 223 B: 222
R: 249 G: 247 B: 250	R: 232 G: 228 B: 230	R: 232 G: 230 B: 231	R: 231 G: 229 B: 230	R: 223 G: 220 B: 221

150,000+ RGB Values



3*3*3 = 27 weights + 1 bias

Convolutions vs. Linear Layers

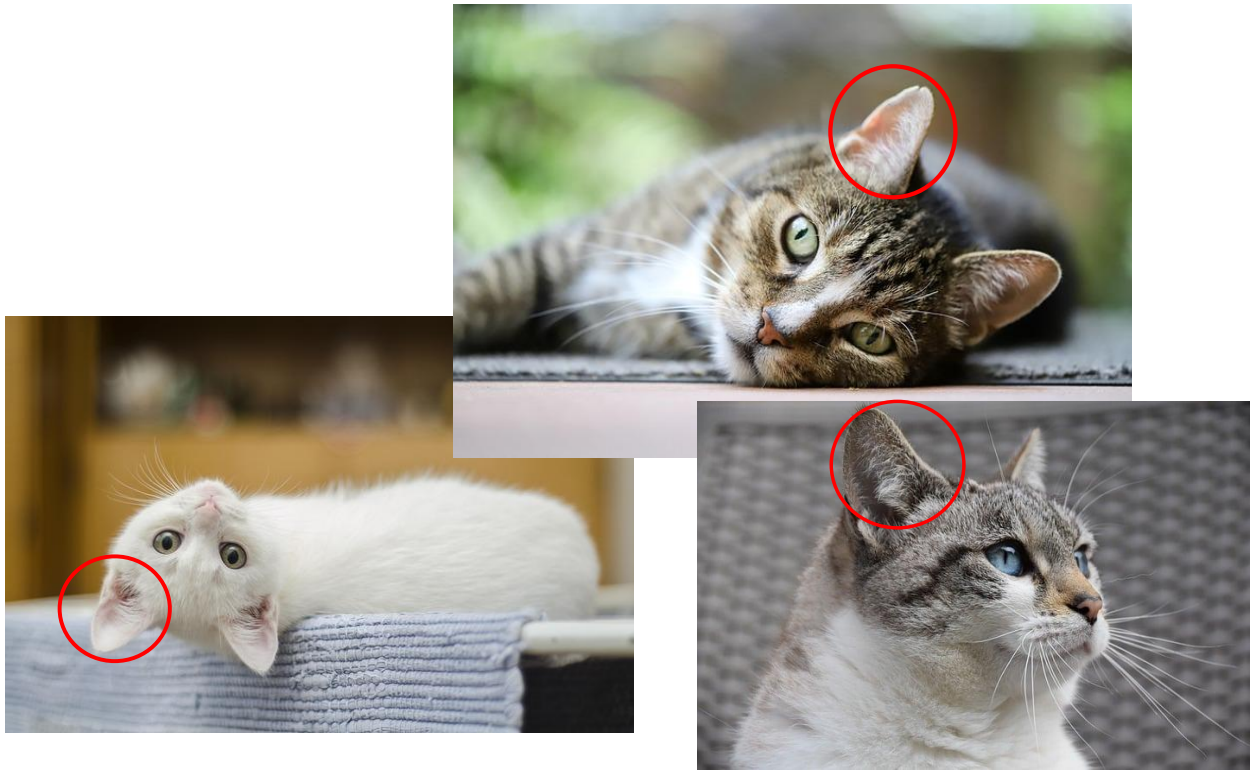
224x224 image

R: 188 G: 179 B: 172	R: 192 G: 178 B: 166	R: 200 G: 180 B: 164	R: 187 G: 164 B: 149	R: 170 G: 149 B: 134
R: 208 G: 201 B: 195	R: 215 G: 197 B: 184		R: 206 G: 185 B: 170	R: 181 G: 161 B: 145
R: 232 G: 228 B: 225				R: 204 G: 192 B: 182
R: 252 G: 249 B: 250	R: 240 G: 234 B: 231		R: 228 G: 223 B: 220	R: 226 G: 223 B: 222
R: 249 G: 247 B: 250	R: 232 G: 228 B: 230	R: 232 G: 230 B: 231	R: 231 G: 229 B: 230	R: 223 G: 220 B: 221

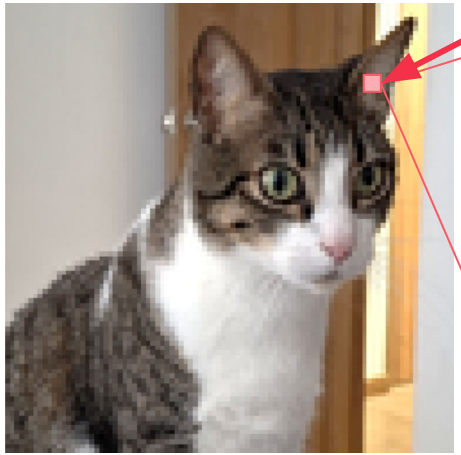
150,000+ RGB Values

-1	0	1
-2	0	2
-1	0	1

Convolutions vs. Linear Layers



Receptive Fields

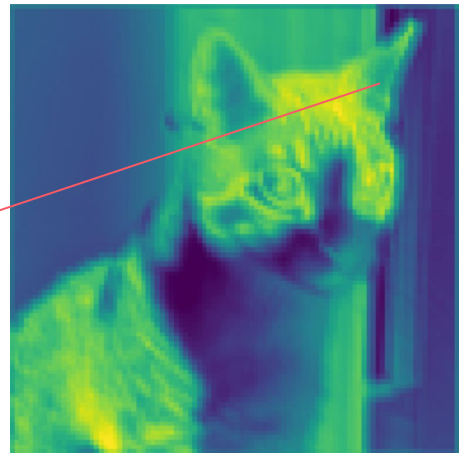


96x96 image

R: 229 G: 220 B: 224	R: 234 G: 228 B: 233	R: 229 G: 224 B: 219
R: 231 G: 223 B: 227	R: 234 G: 227 B: 233	R: 240 G: 234 B: 239
R: 224 G: 216 B: 219	R: 240 G: 232 B: 237	R: 245 G: 239 B: 243

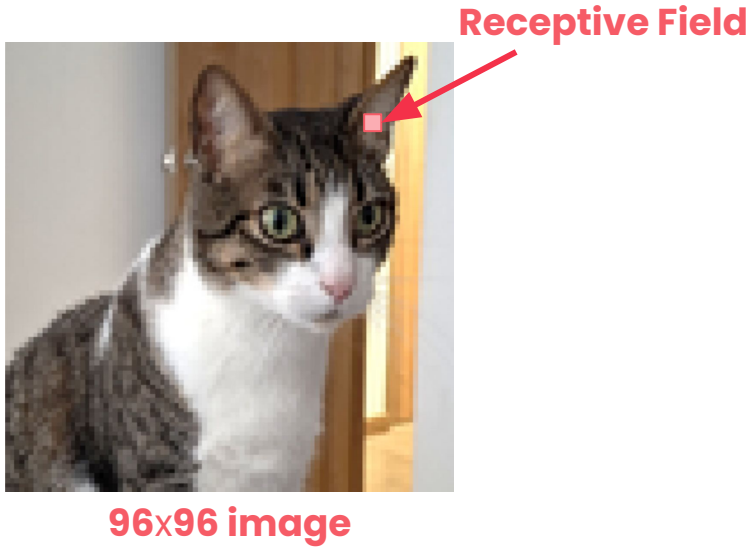
3x3 filter = 9 Pixels

.215



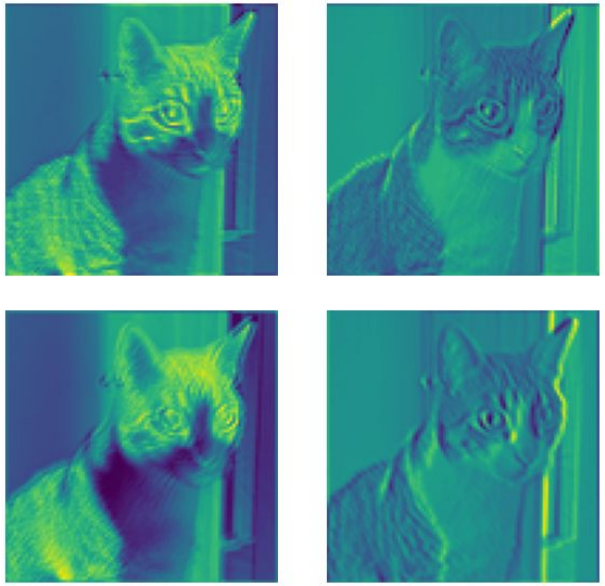
Feature Map
96x96 activations

Receptive Fields



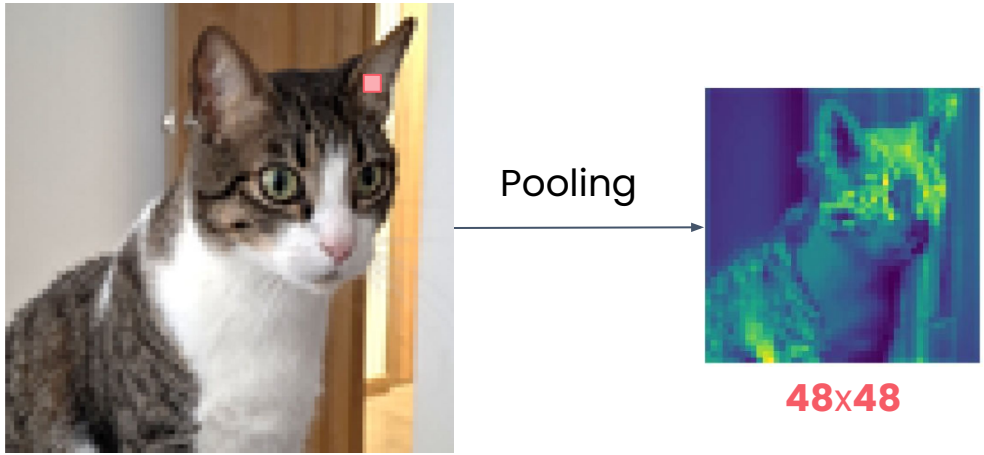
Receptive Fields

96x96 image

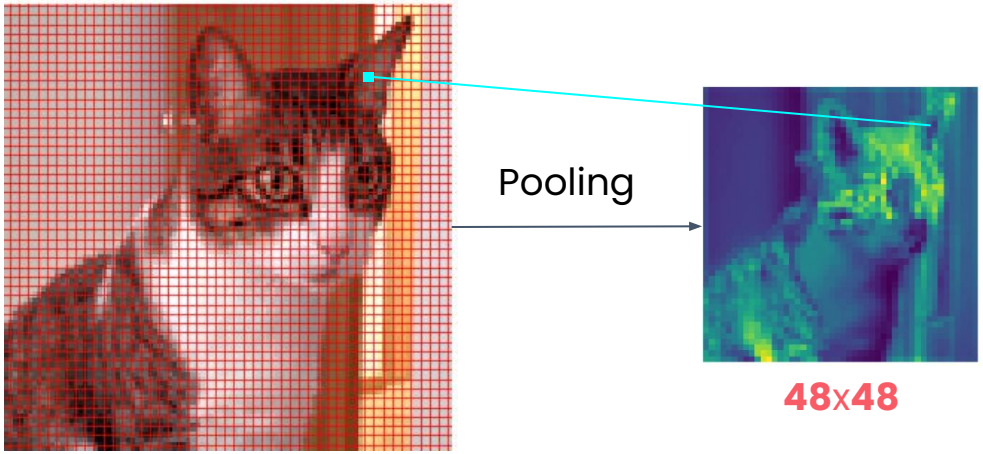


Feature Maps

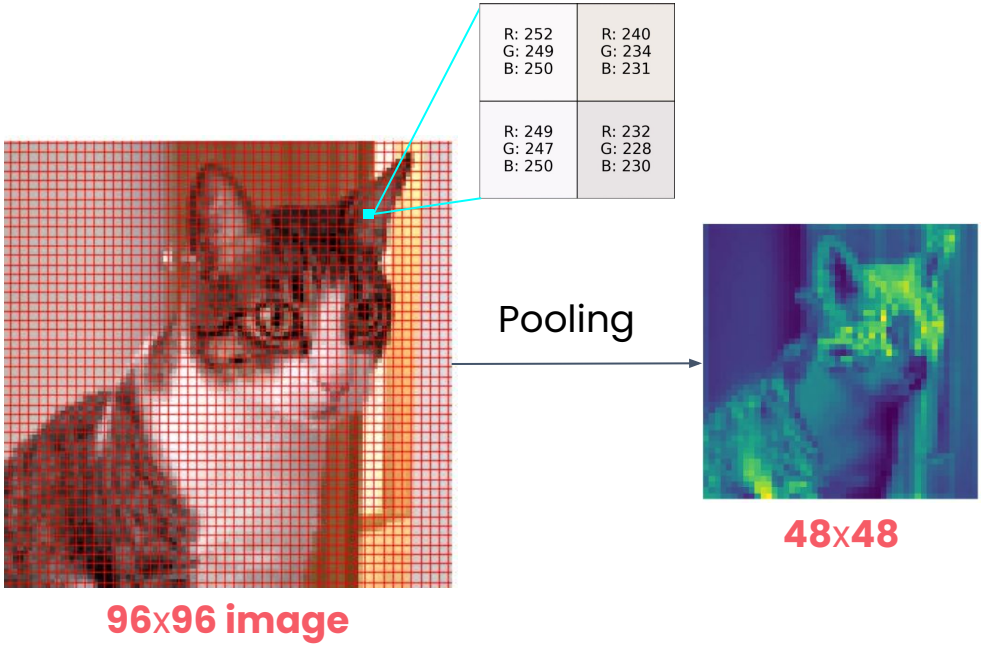
Receptive Fields



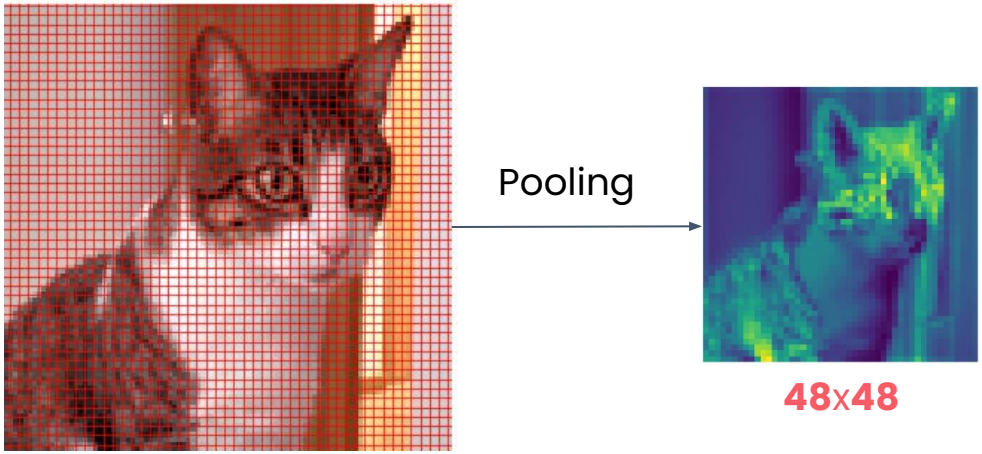
Receptive Fields



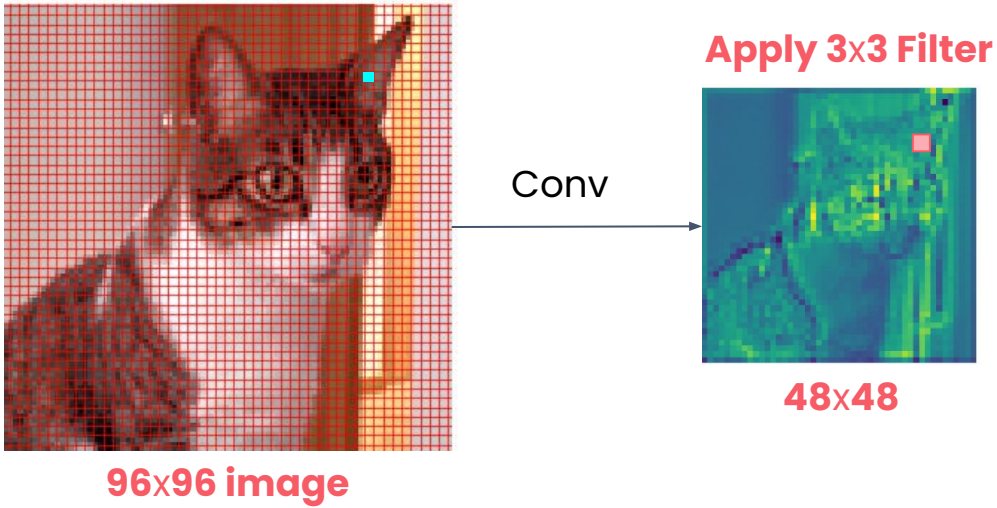
Receptive Fields



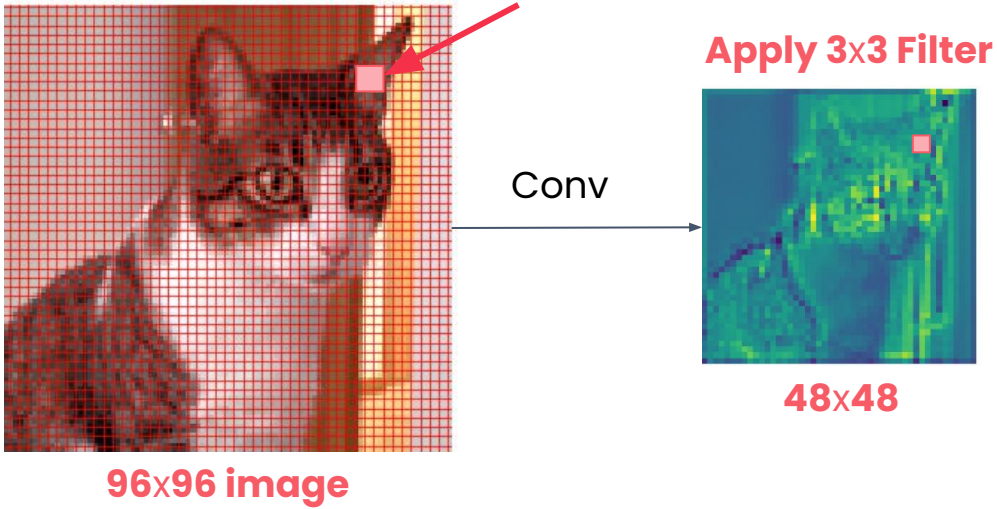
Receptive Fields



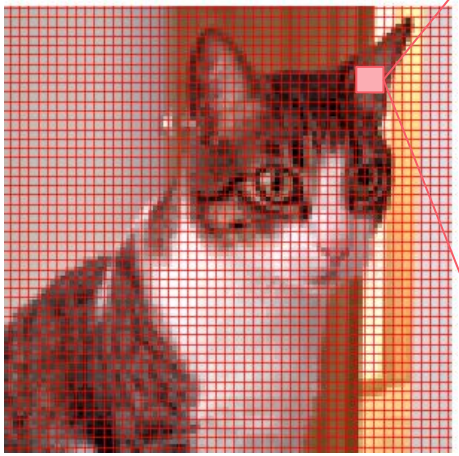
Receptive Fields



Receptive Fields



Receptive Fields

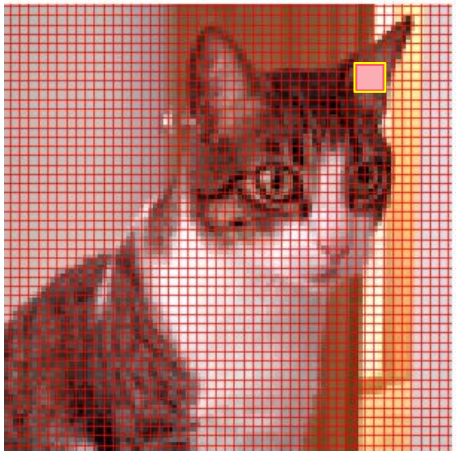


96x96 image

R: 229 G: 220 B: 224	R: 234 G: 228 B: 233	R: 245 G: 243 B: 247	R: 240 G: 239 B: 243	R: 235 G: 235 B: 239	R: 156 G: 153 B: 158
R: 231 G: 223 B: 227	R: 234 G: 227 B: 233	R: 240 G: 234 B: 239	R: 243 G: 241 B: 247	R: 234 G: 234 B: 241	R: 208 G: 207 B: 212
R: 224 G: 216 B: 219	R: 240 G: 232 B: 237	R: 245 G: 239 B: 243	R: 231 G: 229 B: 234	R: 204 G: 203 B: 210	R: 193 G: 193 B: 198
R: 219 G: 206 B: 210	R: 237 G: 228 B: 231	R: 244 G: 237 B: 240	R: 231 G: 226 B: 231	R: 202 G: 199 B: 205	R: 206 G: 206 B: 209
R: 212 G: 194 B: 198	R: 231 G: 215 B: 220	R: 237 G: 222 B: 224	R: 223 G: 211 B: 215	R: 222 G: 219 B: 223	R: 232 G: 232 B: 235
R: 201 G: 178 B: 178	R: 226 G: 202 B: 206	R: 230 G: 207 B: 209	R: 222 G: 204 B: 205	R: 232 G: 227 B: 232	R: 242 G: 242 B: 245

3x3 filter = 36 Pixels

Receptive Fields

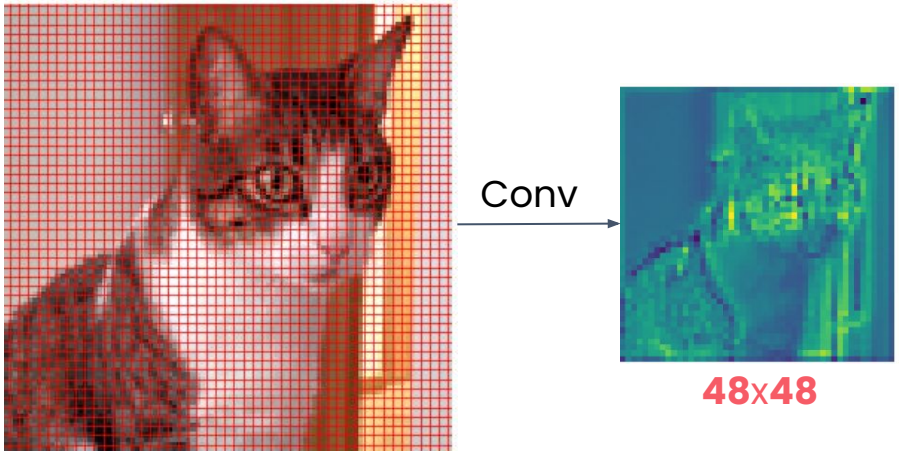


Receptive Fields

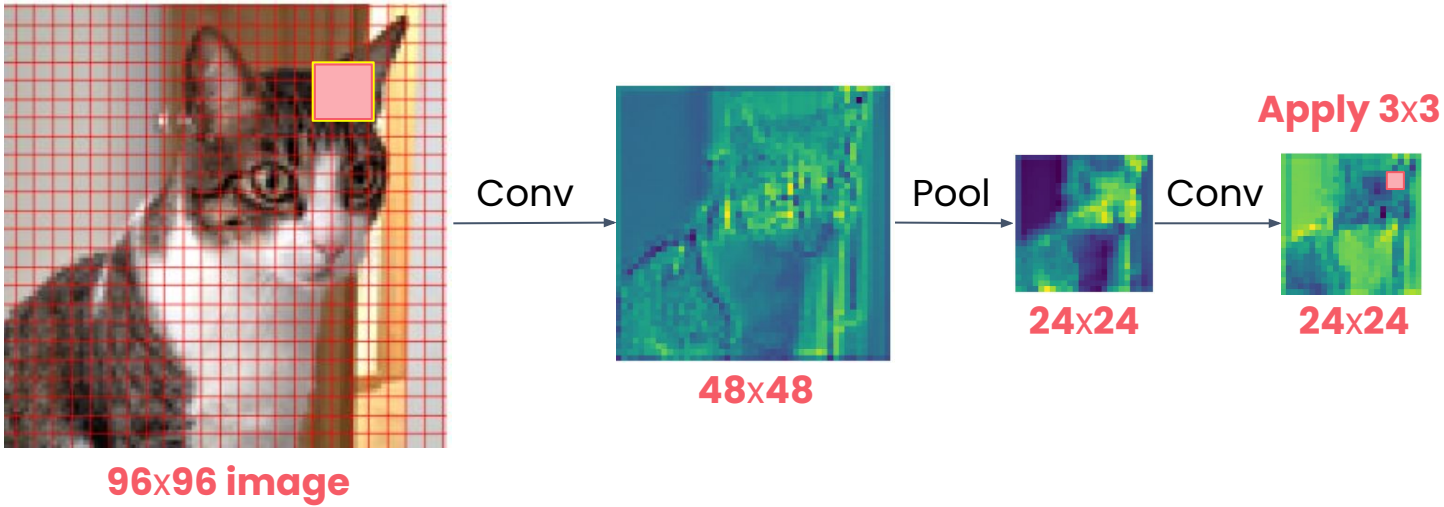
Receptive Field
Before 1st MaxPool: 9 pixels

Receptive Field
After 1st MaxPool: 36 pixels

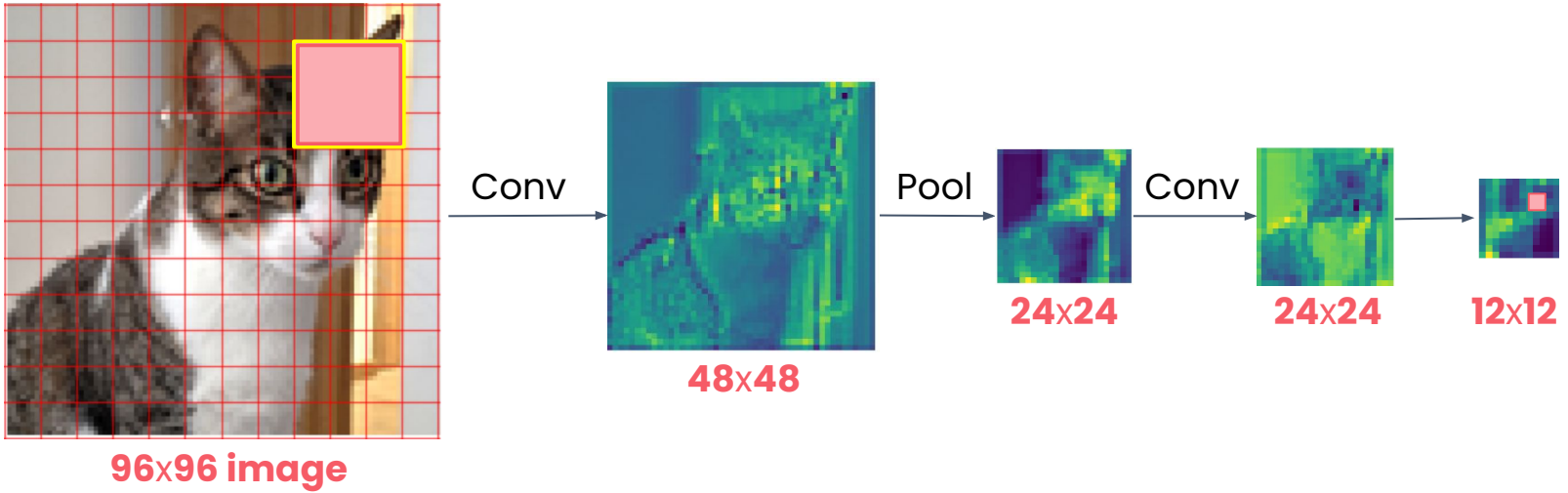
Receptive Fields



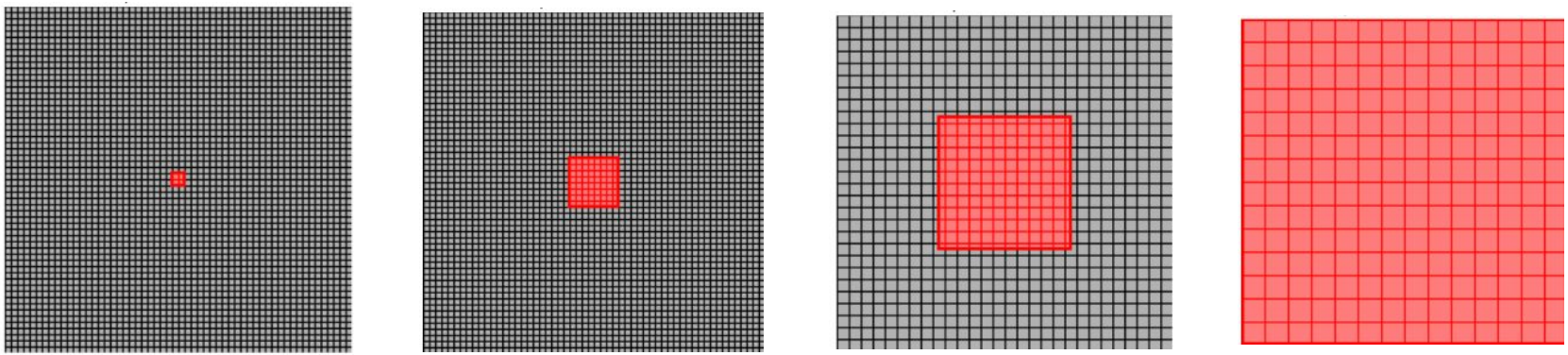
Receptive Fields



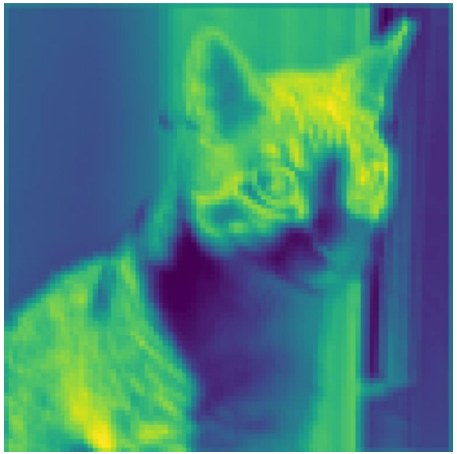
Receptive Fields



Receptive Fields



Receptive Fields

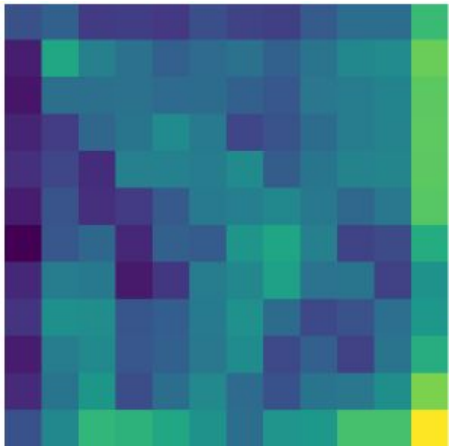


96x96

12x12

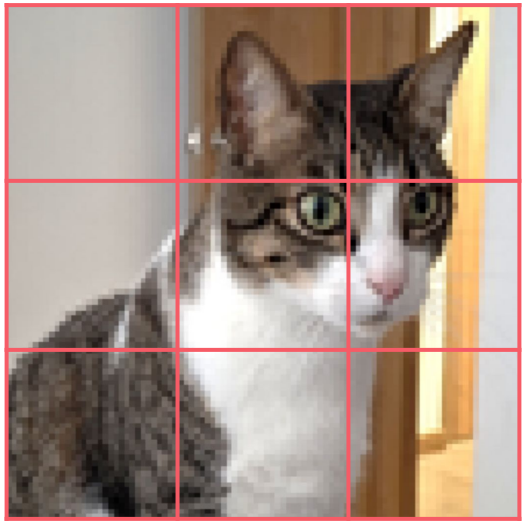
Receptive Fields

96x96



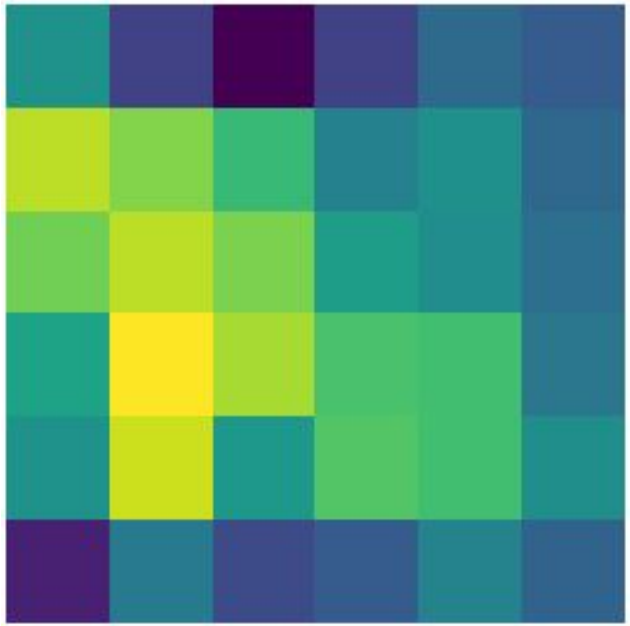
12x12

Receptive Fields



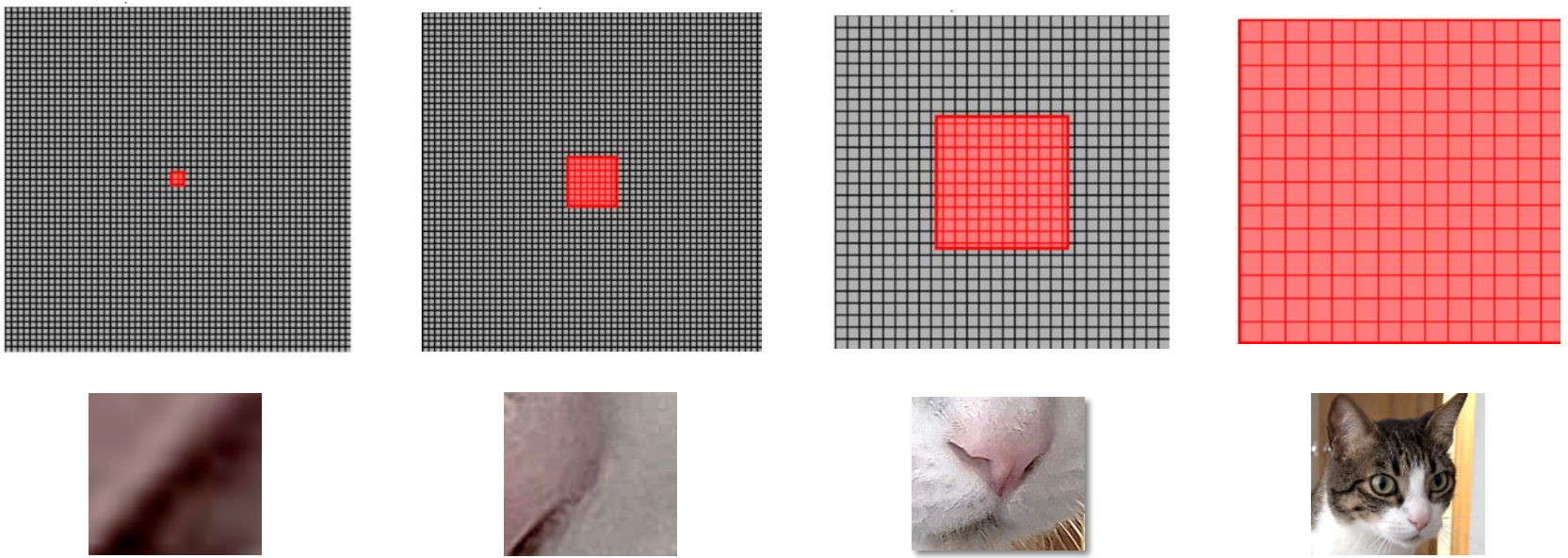
Receptive Fields

This is a cat!



6x6

Receptive Fields

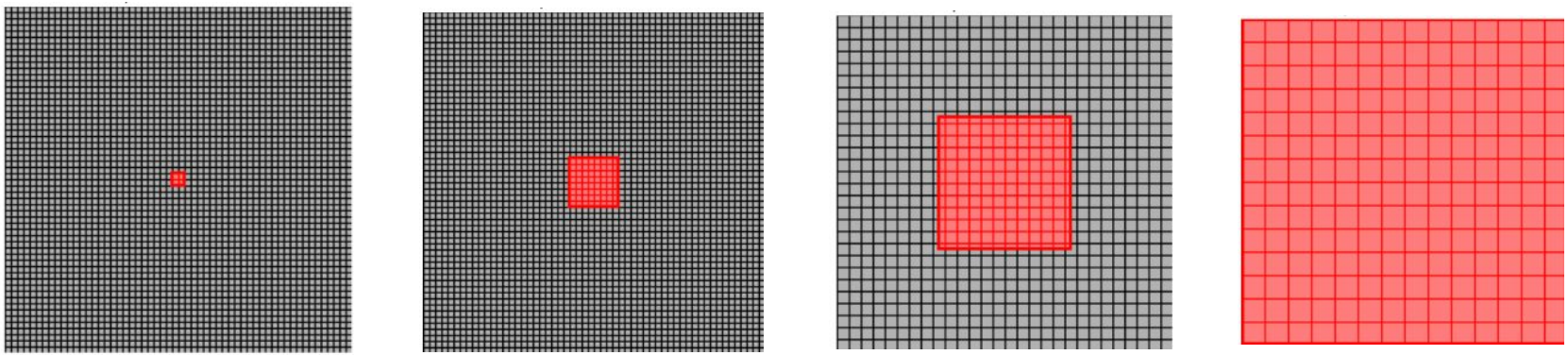


DeepLearning.AI

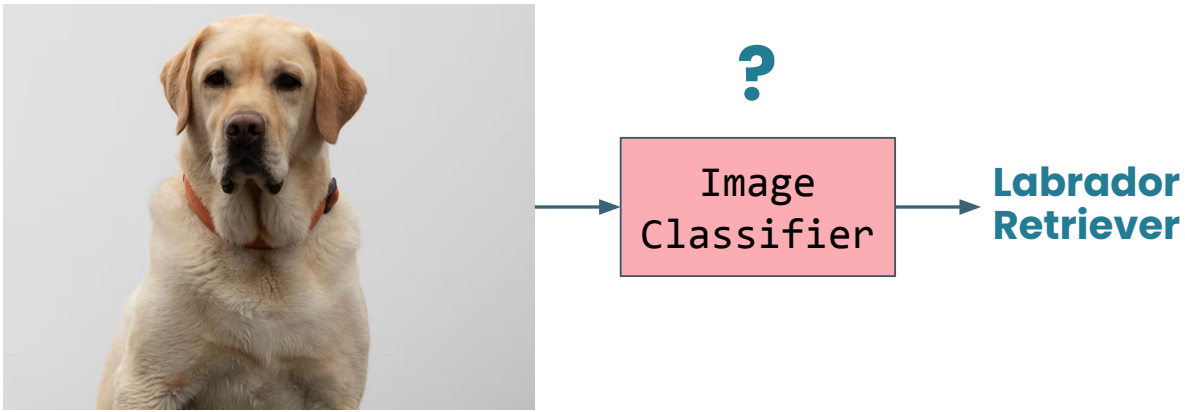
Saliency Maps

Specialized Approaches to Vision

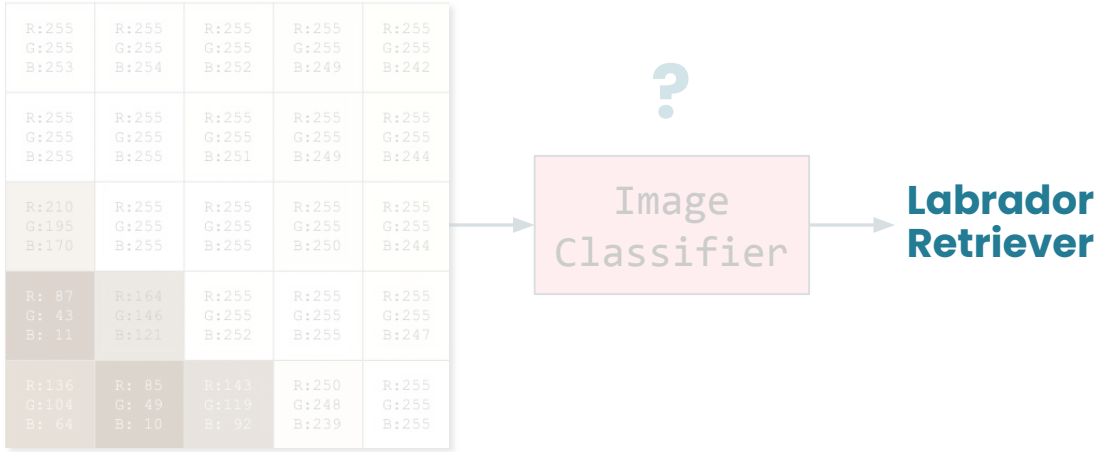
Receptive Fields



Classification



Classification



Interpretability Techniques

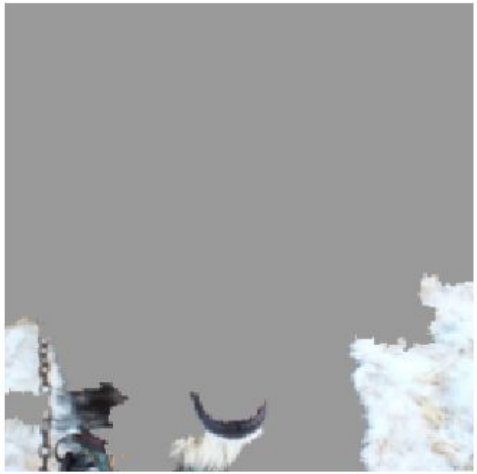
Reveal which parts of an input most affect a models' prediction

Why Interpretability Matters

- **Trust**

Why Interpretability Matters

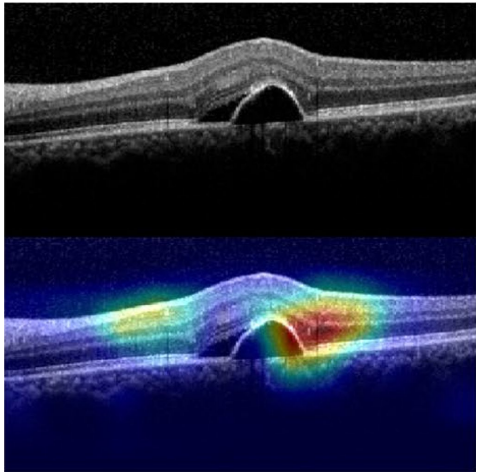
- Trust
- **Debugging**



Ribeiro, Singh & Guestrin (2016)

Why Interpretability Matters

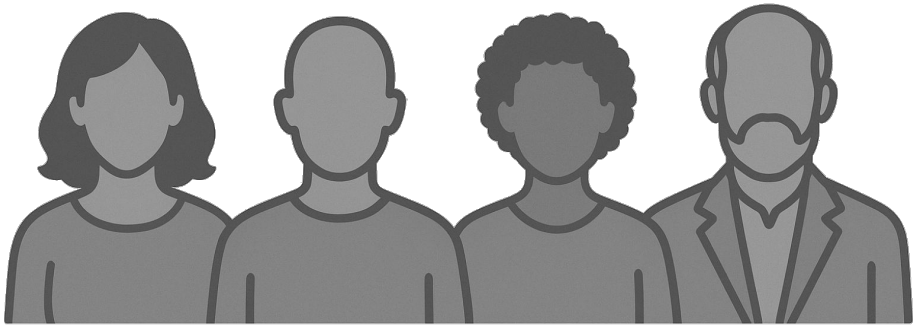
- Trust
- Debugging
- **Discovery**



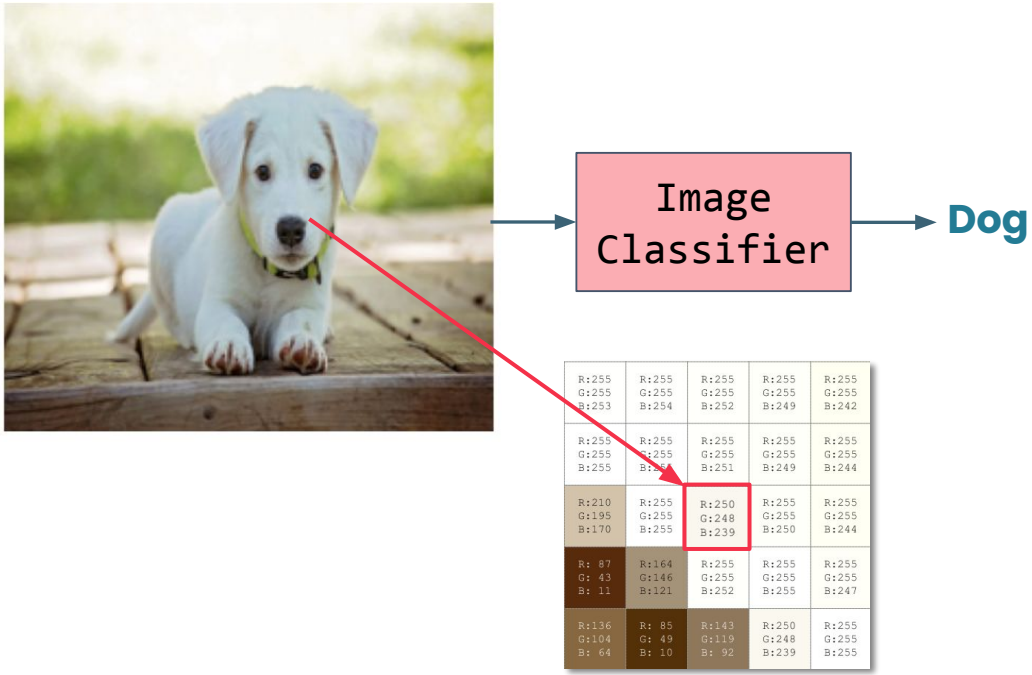
Yoon et al. (2020)

Why Interpretability Matters

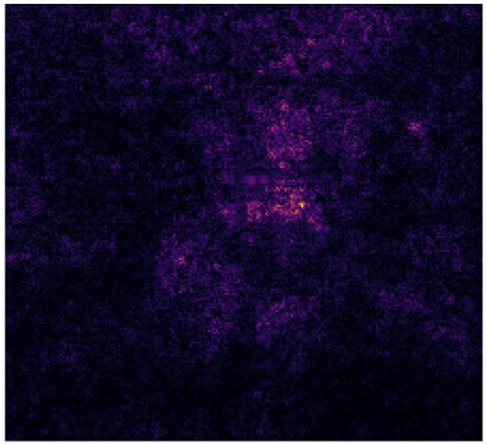
- Trust
- Debugging
- Discovery
- **Ethical Concerns**



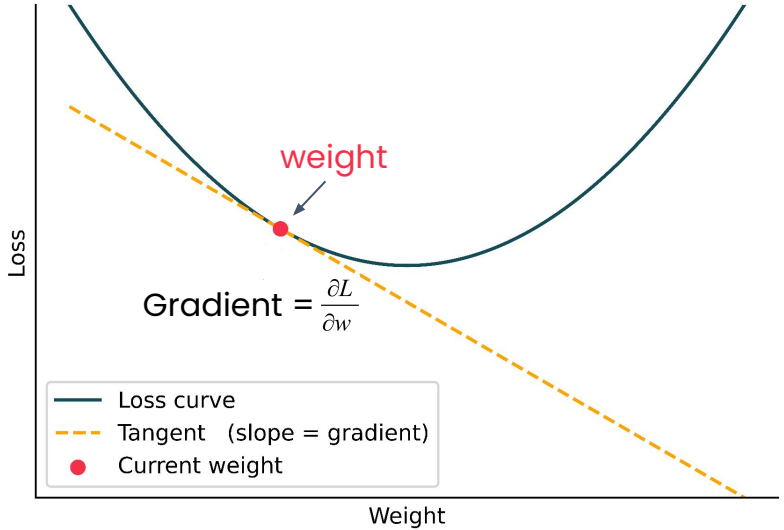
Saliency Maps



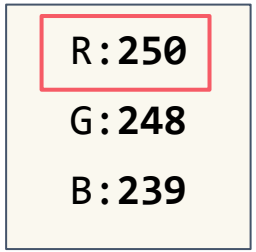
Saliency Maps



Gradients

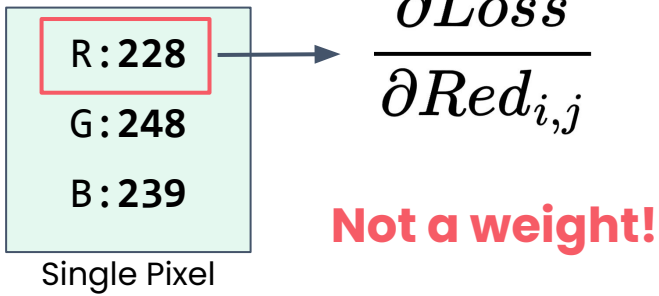

weight
Loss
Gradient = $\frac{\partial L}{\partial w}$
Loss curve
Tangent (slope = gradient)
Current weight
Weight

Gradients on Pixels



Single Pixel

Gradients on Pixels



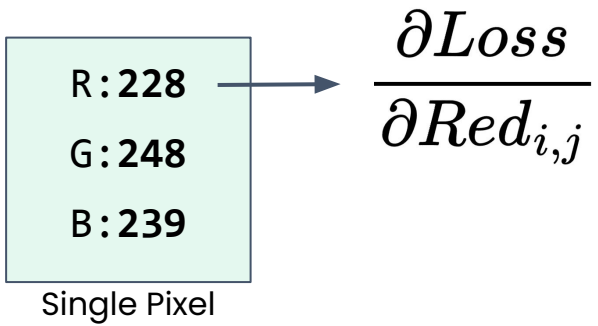
Gradients on Pixels

```
input_image.requires_grad_()
```

Gradients on Pixels

```
input_image.requires_grad_()

output = model(input_image)
output[0, target_class].backward()
```

Gradients on Pixels



Saliency Maps

Saliency Maps

```
input_image = input_image.clone().detach()
input_image.requires_grad_()

output = model(input_image)
model.zero_grad()
output[0, target_class].backward()

gradients = input_image.grad.data[0]

saliency_map = torch.abs(gradients).sum(dim=0).vcpu().numpy()

saliency_map = (saliency_map - saliency_map.min()) /
               (saliency_map.max() - saliency_map.min() + 1e-8)
```

Saliency Maps

```
input_image = input_image.clone().detach()
input_image.requires_grad_()

output = model(input_image)
model.zero_grad()
output[0, target_class].backward()

gradients = input_image.grad.data[0]

saliency_map = torch.abs(gradients).sum(dim=0).vcpu().numpy()

saliency_map = (saliency_map - saliency_map.min()) /
               (saliency_map.max() - saliency_map.min() + 1e-8)
```

Saliency Maps

```
input_image = input_image.clone().detach()
input_image.requires_grad_()

output = model(input_image)
model.zero_grad()
output[0, target_class].backward()

gradients = input_image.grad.data[0]

saliency_map = torch.abs(gradients).sum(dim=0).vcpu().numpy()

saliency_map = (saliency_map - saliency_map.min()) /
               (saliency_map.max() - saliency_map.min() + 1e-8)
```

Saliency Maps

```
input_image = input_image.clone().detach()
input_image.requires_grad_()

output = model(input_image)
model.zero_grad()
output[0, target_class].backward()

gradients = input_image.grad.data[0]

saliency_map = torch.abs(gradients).sum(dim=0).vcpu().numpy()

saliency_map = (saliency_map - saliency_map.min()) /
               (saliency_map.max() - saliency_map.min() + 1e-8)
```

Saliency Maps

```
input_image = input_image.clone().detach()
input_image.requires_grad_()

output = model(input_image)
model.zero_grad()
output[0, target_class].backward()

gradients = input_image.grad.data[0]

saliency_map = torch.abs(gradients).sum(dim=0).vcpu().numpy()

saliency_map = (saliency_map - saliency_map.min()) /
               (saliency_map.max() - saliency_map.min() + 1e-8)
```

Saliency Maps

```
# Load pre-trained ResNet50 model
model = models.resnet50(weights=models.ResNet50_Weights.IMAGENET1K_V1)
model.eval()
model.to(device)
```

Saliency Maps

```
# Load pre-trained ResNet50 model
model = models.resnet50(weights=models.ResNet50_Weights.IMAGENET1K_V1)
model.eval()
model.to(device)
```

Saliency Maps

```
# Load pre-trained ResNet50 model
model = models.resnet50(weights=models.ResNet50_Weights.IMAGENET1K_V1)
model.eval()
model.to(device)
```

Saliency Maps

```
# Load pre-trained ResNet50 model
model = models.resnet50(weights=models.ResNet50_Weights.IMAGENET1K_V1)
model.eval()
model.to(device)
```

Saliency Maps

```
img_path = 'images/dog.jpg'

transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])

if not os.path.exists(img_path):
    raise FileNotFoundError("Please provide a valid path to an image.")

img_pil = load_image(img_path)
img_tensor = transform(img_pil).unsqueeze(0).to(device)
```

Saliency Maps

```
img_path = 'images/dog.jpg'

transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])

if not os.path.exists(img_path):
    raise FileNotFoundError("Please provide a valid path to an image.")

img_pil = load_image(img_path)
img_tensor = transform(img_pil).unsqueeze(0).to(device)
```

Saliency Maps

```
img_path = 'images/dog.jpg'

transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])

if not os.path.exists(img_path):
    raise FileNotFoundError("Please provide a valid path to an image.")

img_pil = load_image(img_path)
img_tensor = transform(img_pil).unsqueeze(0).to(device)
```

Saliency Maps

```
img_path = 'images/dog.jpg'

transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])

if not os.path.exists(img_path):
    raise FileNotFoundError("Please provide a valid path to an image.")

img_pil = load_image(img_path)
img_tensor = transform(img_pil).unsqueeze(0).to(device)
```

Saliency Maps

```
img_path = 'images/dog.jpg'

transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])

if not os.path.exists(img_path):
    raise FileNotFoundError("Please provide a valid path to an image.")

img_pil = load_image(img_path)
img_tensor = transform(img_pil).unsqueeze(0).to(device)
```

Saliency Maps

```
def compute_saliency_map(model, input_image, target_class=None):
    . . .
```

Saliency Maps

```
def compute_saliency_map(model, input_image, target_class=None):
    . . .
```

Saliency Maps

```
def compute_saliency_map(model, input_image, target_class=None):
    . . .
```

Saliency Maps

```
def compute_saliency_map(model, input_image, target_class=None):
    . . .
```

Saliency Maps

```
def compute_saliency_map(model, input_image, target_class=None):

    input_image = input_image.clone().detach()
    input_image.requires_grad_()
```

Saliency Maps

```
def compute_saliency_map(model, input_image, target_class=None):

    input_image = input_image.clone().detach()
    input_image.requires_grad_()
```

leaf tensor

Saliency Maps

```
def compute_saliency_map(model, input_image, target_class=None):

    input_image = input_image.clone().detach()
    input_image.requires_grad_()
```

Saliency Maps

```
def compute_saliency_map(model, input_image, target_class=None):

    input_image = input_image.clone().detach()
    input_image.requires_grad_()

    output = model(input_image)
```

Saliency Maps

```
def compute_saliency_map(model, input_image, target_class=None):

    input_image = input_image.clone().detach()
    input_image.requires_grad_()

    output = model(input_image)
    model.zero_grad()
    output[0, target_class].backward()
```

Saliency Maps

```
def compute_saliency_map(model, input_image, target_class=None):

    input_image = input_image.clone().detach()
    input_image.requires_grad_()

    output = model(input_image)
    model.zero_grad()
    output[0, target_class].backward()
```

Saliency Maps

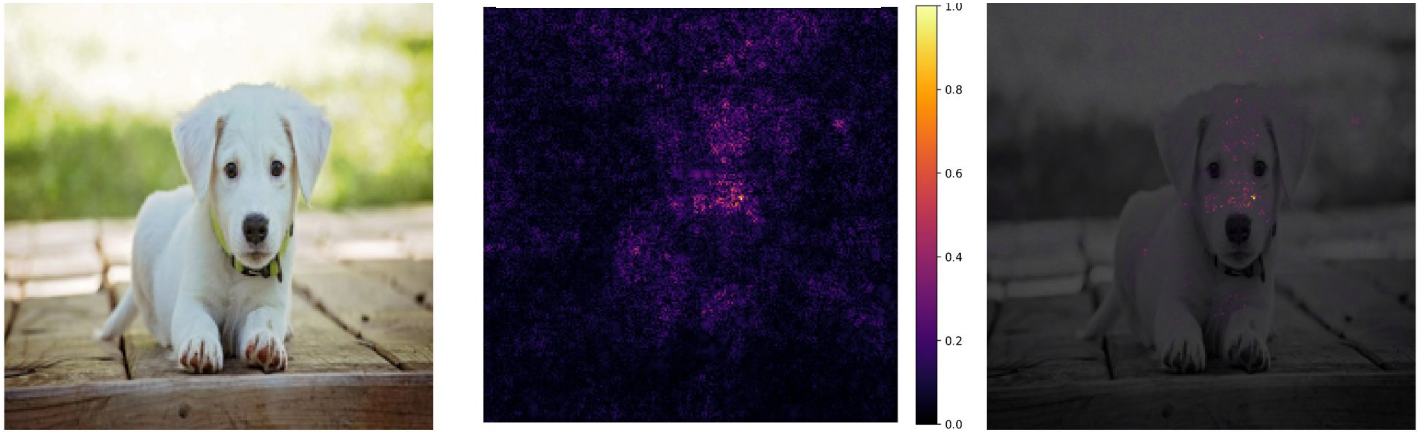
```
def compute_saliency_map(model, input_image, target_class=None):

    input_image = input_image.clone().detach()
    input_image.requires_grad_()

    output = model(input_image)
    model.zero_grad()
    output[0, target_class].backward()

    gradients = input_image.grad.data[0]
```

Saliency Maps

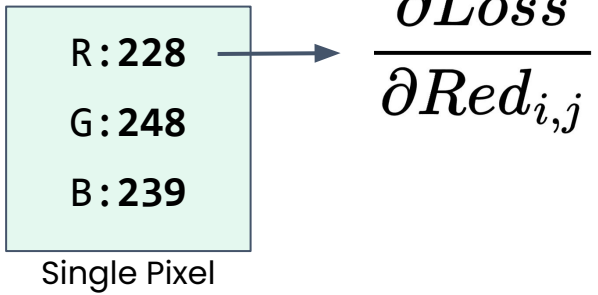
```
def compute_saliency_map(model, input_image, target_class=None):

    input_image = input_image.clone().detach()
    input_image.requires_grad_()

    output = model(input_image)
    model.zero_grad()
    output[0, target_class].backward()

    gradients = input_image.grad.data[0]

    saliency_map = torch.abs(gradients).sum(dim=0).vcpu().numpy()

    saliency_map = (saliency_map - saliency_map.min()) /
                   (saliency_map.max() - saliency_map.min() + 1e-8)
```

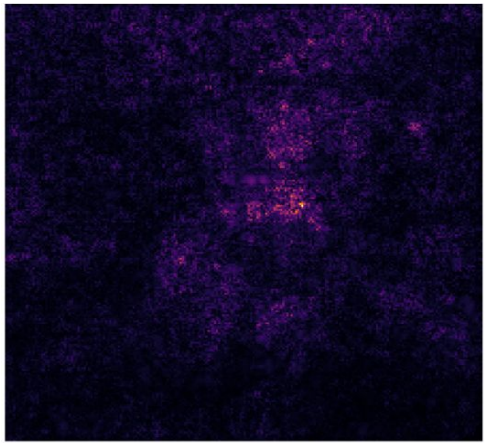
Saliency Maps

```
def visualize_saliency(img_display, saliency_map pred_class, pred_score, title):
    . . .
```



Gradients on Pixels



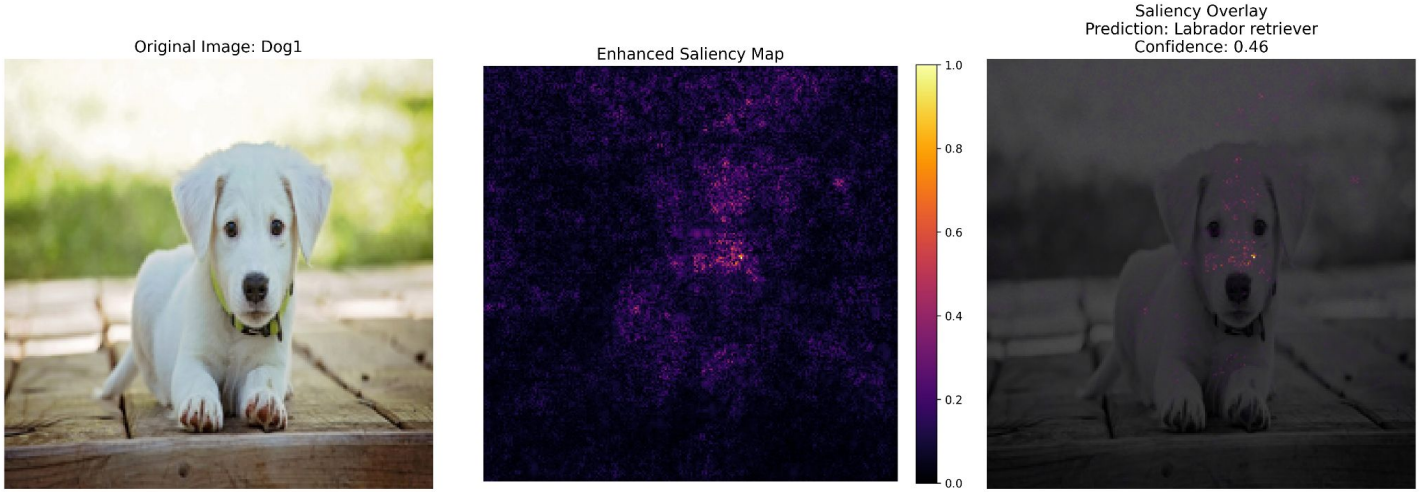


DeepLearning.AI

Class Activation Maps

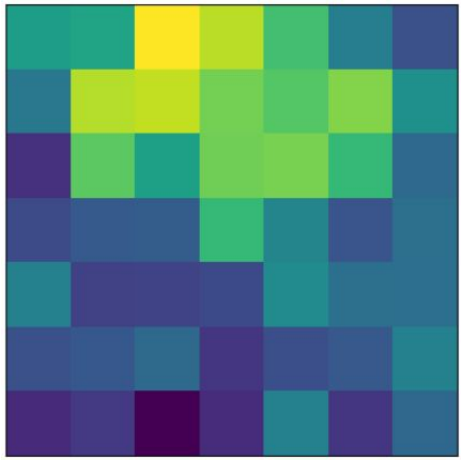
Specialized Approaches to Vision

Saliency Maps



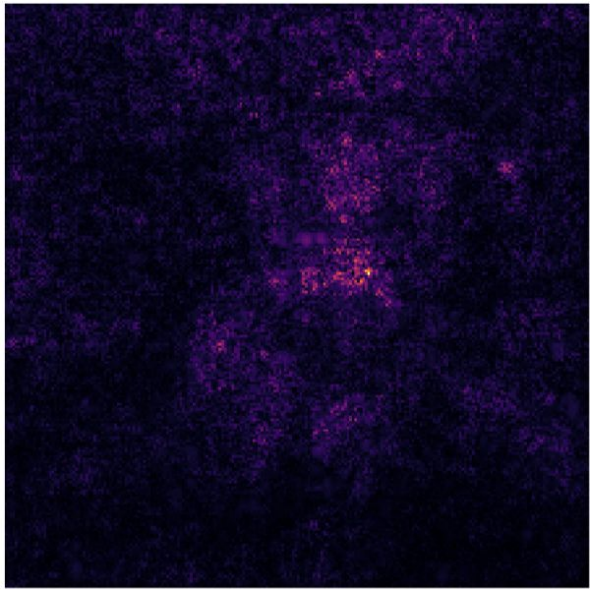
Feature Maps

224x224 image

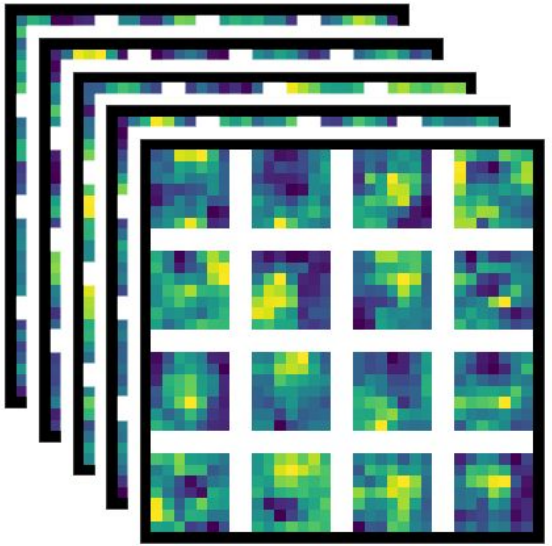


7x7 Feature Map

CAMs

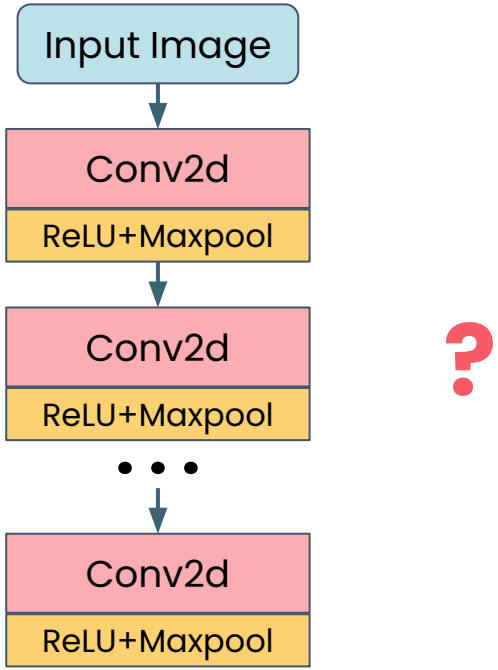


Saliency Maps:
Which pixels matter most?



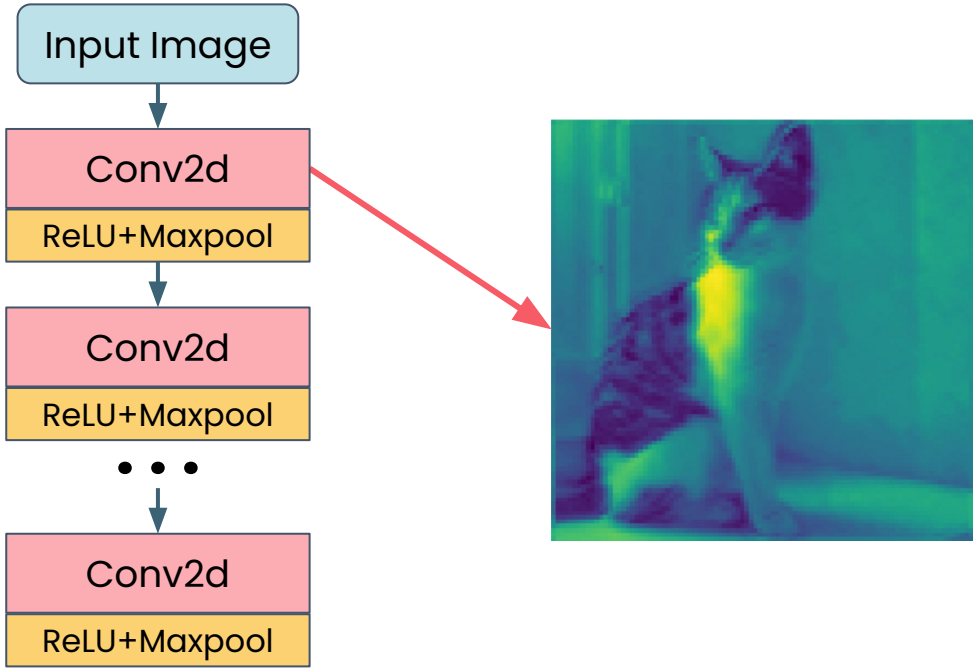
Class Activation Maps (CAMs):
Which feature maps matter most?

Feature Maps

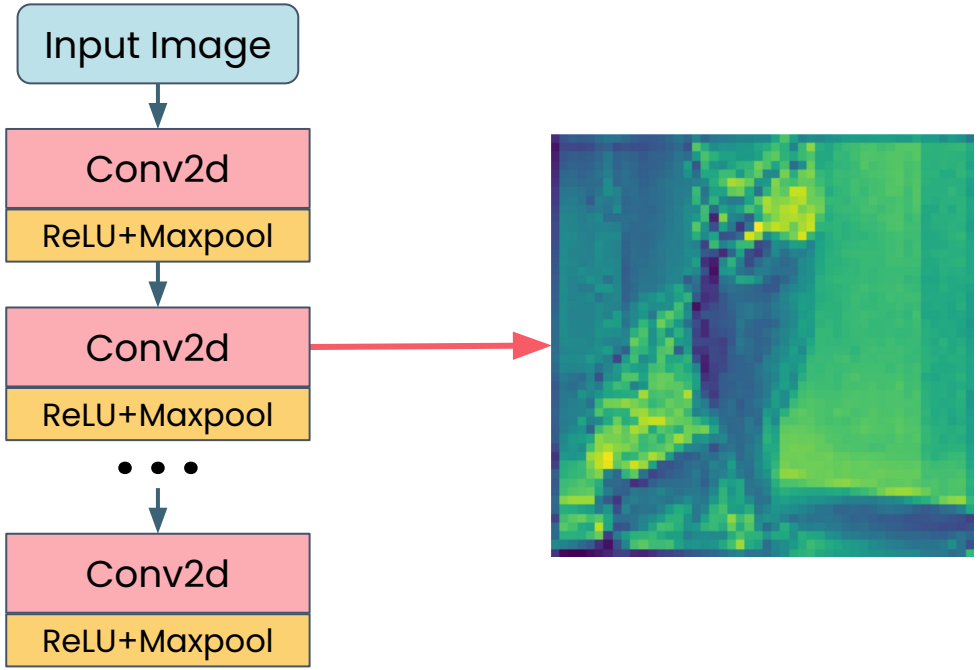


?

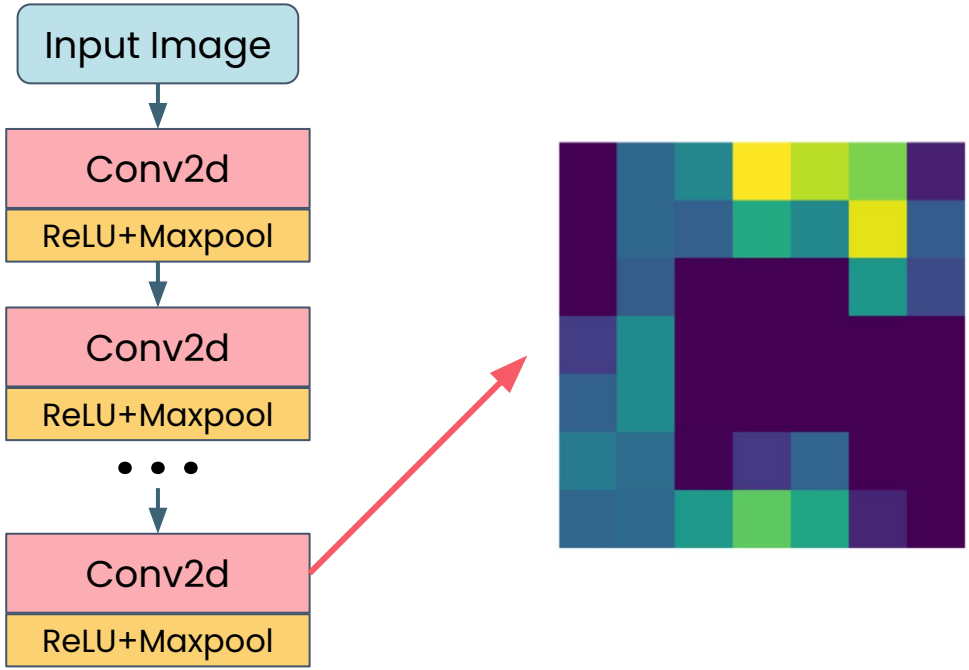
Feature Maps



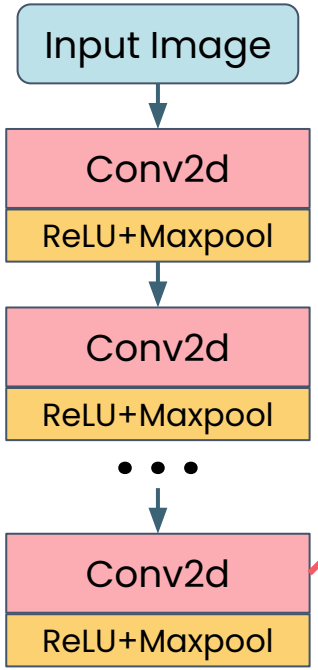
Feature Maps

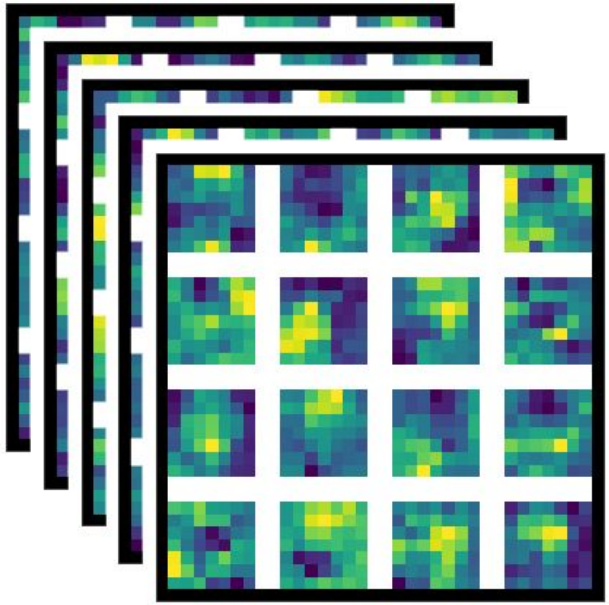


Feature Maps



Feature Maps





2048 Feature Maps
7x7

224x224 Image

Class Activation Maps

GradCAM

```
# Load pretrained ResNet50 model and class labels
model = models.resnet50(weights=models.ResNet50_Weights.IMAGENET1K_V1)
model.eval()
model.to(device)
```

GradCAM

```
# Load pretrained ResNet50 model and class labels
model = models.resnet50(weights=models.ResNet50_Weights.IMAGENET1K_V1)
model.eval()
model.to(device)
```

GradCAM

```
# Load pretrained ResNet50 model and class labels
model = models.resnet50(weights=models.ResNet50_Weights.IMAGENET1K_V1)
model.eval()
model.to(device)


class GradCAM:
    def __init__(self, model, target_layer):
        . . .
```

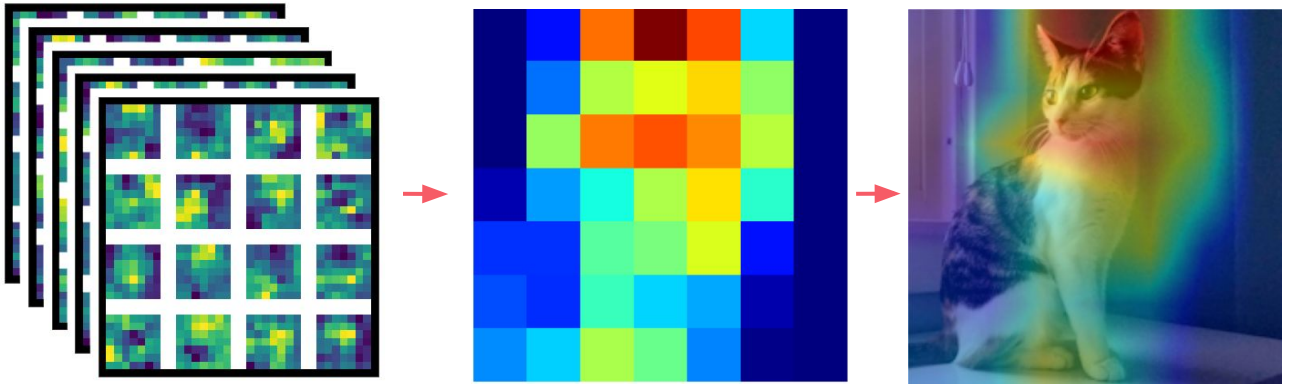
GradCAM

```
# Load pretrained ResNet50 model and class labels
model = models.resnet50(weights=models.ResNet50_Weights.IMAGENET1K_V1)
model.eval()
model.to(device)


class GradCAM:
    def __init__(self, model, target_layer):
        . . .
```

GradCAM

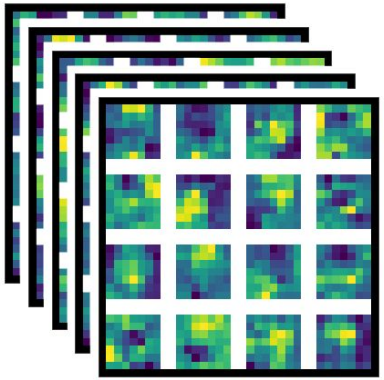
```
# Load pretrained ResNet50 model and class labels
model = models.resnet50(weights=models.ResNet50_Weights.IMAGENET1K_V1)
model.eval()
model.to(device)


class GradCAM:
    def __init__(self, model, target_layer):
        . . .
```

GradCAM



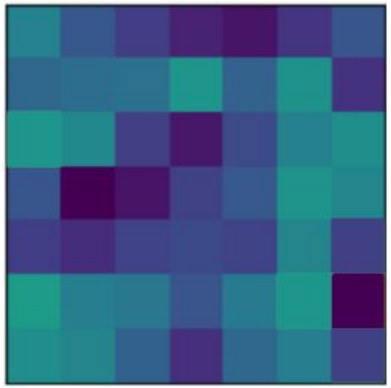
2048 Feature Maps
7x7

GradCAM

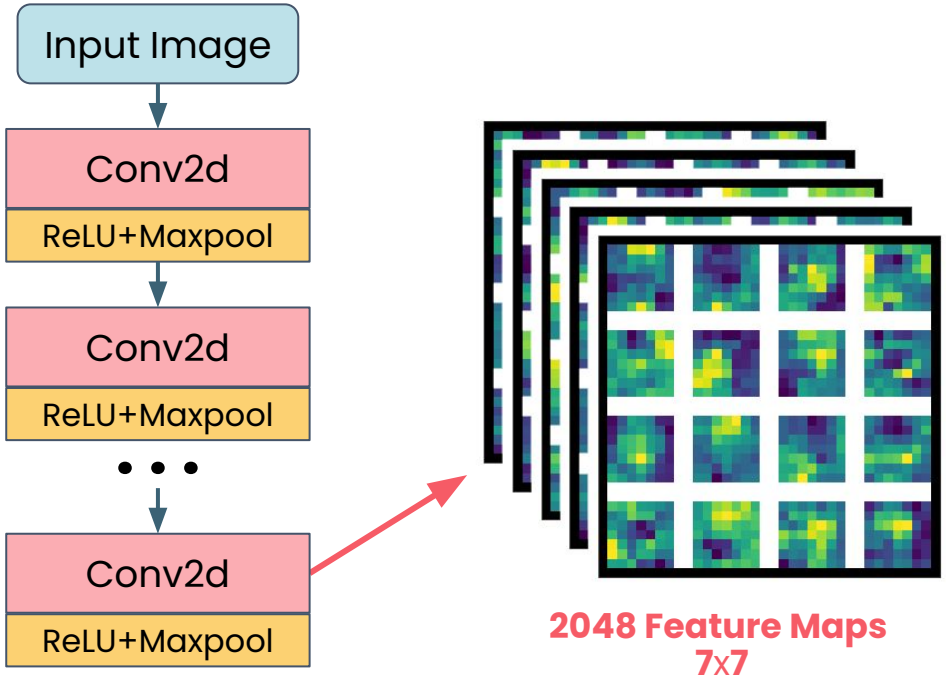


2048 Feature Maps
7x7

Feature Map #274



GradCAM


Input Image
Conv2d
ReLU+Maxpool
Conv2d
ReLU+Maxpool
Conv2d
ReLU+Maxpool
2048 Feature Maps
7x7
Intermediate Tensors
=
No saved gradients

Hooks

```
# "Hook" (register callback) into target layer for forward activations
target_layer.register_forward_hook(self.save_activation)

# Hook into the backward pass to save the gradients
target_layer.register_full_backward_hook(self.save_gradient)
```

Hooks

```
# "Hook" (register callback) into target layer for forward activations
target_layer.register_forward_hook(self.save_activation)

# Hook into the backward pass to save the gradients
target_layer.register_full_backward_hook(self.save_gradient)
```

Hooks

```
# "Hook" (register callback) into target layer for forward activations
target_layer.register_forward_hook(self.save_activation)

# Hook into the backward pass to save the gradients
target_layer.register_full_backward_hook(self.save_gradient)
```

Hooks

```
# "Hook" (register callback) into target layer for forward activations
target_layer.register_forward_hook(self.save_activation)

# Hook into the backward pass to save the gradients
target_layer.register_full_backward_hook(self.save_gradient)
```

Hooks: Forward Pass

```
class GradCAM(nn.Module):
  def __call__(self, x, class_idx=None):

      # Forward Pass
      output = self.model(x)
```

```
# Automatically called after the forward pass
def save_activation(self, module, input, output):
    # .detach() to avoid linking this to computational graph
    self.activations = output.detach()
```

Hooks: Forward Pass

```
class GradCAM(nn.Module):
  def __call__(self, x, class_idx=None):

    # Forward Pass
    output = self.model(x)
```

```
# Automatically called after the forward pass
def save_activation(self, module, input, output):
    # .detach() to avoid linking this to computational graph
    self.activations = output.detach()
```

Hooks: Forward Pass

```
class GradCAM(nn.Module):
  def __call__(self, x, class_idx=None):

    # Forward Pass
    output = self.model(x)
```

```
# Automatically called after the forward pass
def save_activation(self, module, input, output):
    # .detach() to avoid linking this to computational graph
    self.activations = output.detach()
```

Hooks: Backward Pass

```
class GradCAM(nn.Module):
  def __call__(self, x, class_idx=None):
      . . .
      # Backward Pass
      self.model.zero_grad()
      one_hot = torch.zeros_like(output)
      one_hot[0, class_idx] = 1
      output.backward(gradient=one_hot, retain_graph=True)
```

```
# Automatically called after the backward pass
def save_gradients(self, module, grad_input, grad_output):
    # grad_output is a tuple; the gradient we care about is [0]
    self.gradients = grad_output[0].detach()
```

Hooks: Backward Pass

```
class GradCAM(nn.Module):
  def __call__(self, x, class_idx=None):
      . . .
      # Backward Pass
      self.model.zero_grad()
      one_hot = torch.zeros_like(output)
      one_hot[0, class_idx] = 1
      output.backward(gradient=one_hot, retain_graph=True)
```

```
# Automatically called after the backward pass
def save_gradients(self, module, grad_input, grad_output):
    # grad_output is a tuple; the gradient we care about is [0]
    self.gradients = grad_output[0].detach()
```

Hooks: Backward Pass

```
class GradCAM(nn.Module):
  def __call__(self, x, class_idx=None):
      . . .
      # Backward Pass
      self.model.zero_grad()
      one_hot = torch.zeros_like(output)
      one_hot[0, class_idx] = 1
      output.backward(gradient=one_hot, retain_graph=True)
```

```
# Automatically called after the backward pass
def save_gradients(self, module, grad_input, grad_output):
    # grad_output is a tuple; the gradient we care about is [0]
    self.gradients = grad_output[0].detach()
```

Hooks: Backward Pass

```
class GradCAM(nn.Module):
  def __call__(self, x, class_idx=None):
      . . .
      # Backward Pass
      self.model.zero_grad()
      one_hot = torch.zeros_like(output)
      one_hot[0, class_idx] = 1
      output.backward(gradient=one_hot, retain_graph=True)
```

```
# Automatically called after the backward pass
def save_gradients(self, module, grad_input, grad_output):
    # grad_output is a tuple; the gradient we care about is [0]
    self.gradients = grad_output[0].detach()
```

Hooks: Backward Pass

```
class GradCAM(nn.Module):
  def __call__(self, x, class_idx=None):
      . . .
      # Backward Pass
      self.model.zero_grad()
      one_hot = torch.zeros_like(output)
      one_hot[0, class_idx] = 1
      output.backward(gradient=one_hot, retain_graph=True)
```

```
# Automatically called after the backward pass
def save_gradients(self, module, grad_input, grad_output):
    # grad_output is a tuple; the gradient we care about is [0]
    self.gradients = grad_output[0].detach()
```

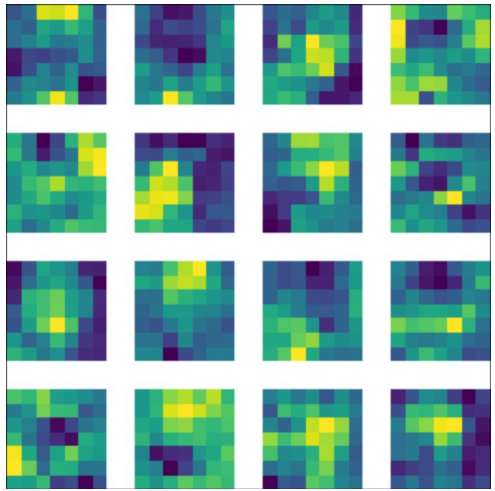
Hooks: Backward Pass

```
class GradCAM(nn.Module):
  def __call__(self, x, class_idx=None):
      . . .
      # Backward Pass
      self.model.zero_grad()
      one_hot = torch.zeros_like(output)
      one_hot[0, class_idx] = 1
      output.backward(gradient=one_hot, retain_graph=True)
```

```
# Automatically called after the backward pass
def save_gradients(self, module, grad_input, grad_output):
    # grad_output is a tuple; the gradient we care about is [0]
    self.gradients = grad_output[0].detach()
```

Activations and Gradients

Activations



2048 Feature Maps
7x7

Gradients

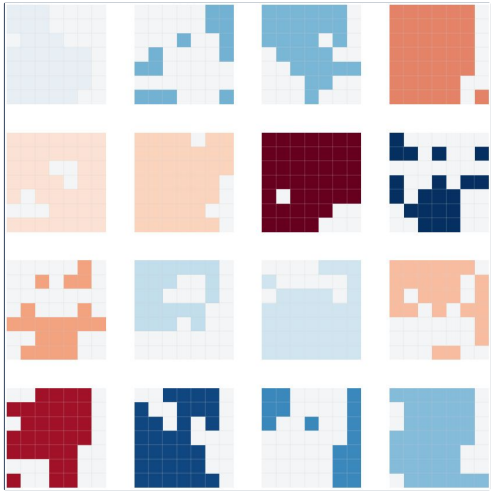
2048 Gradient Maps
7x7

Activations and Gradients

Activations

2048 Feature Maps
7x7

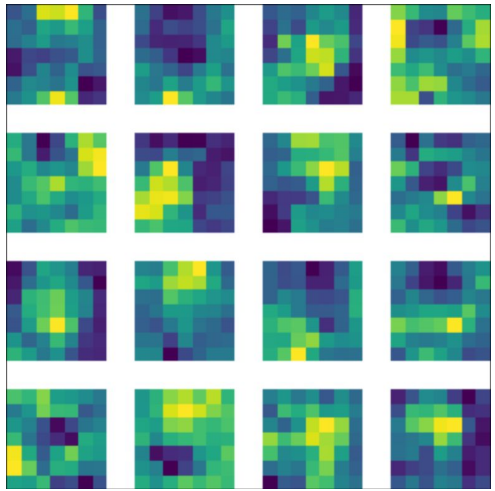
Gradients



2048 Gradient Maps
7x7

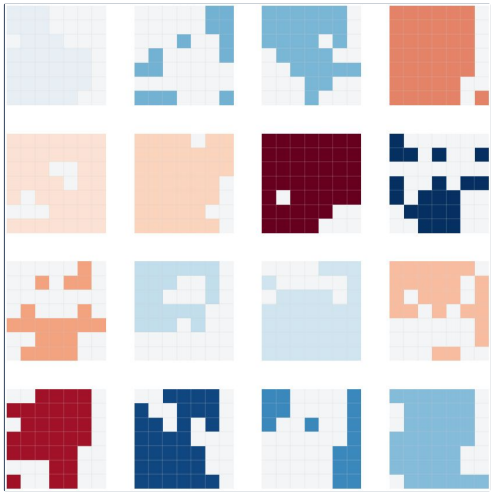
Activations and Gradients

Activations



2048 Feature Maps
7x7

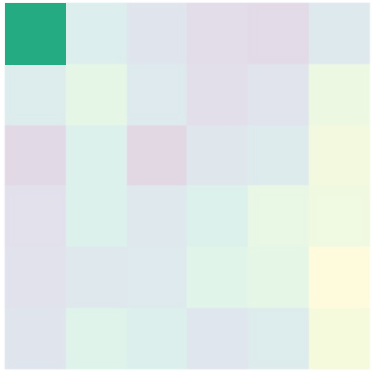
Gradients



2048 Gradient Maps
7x7

Activations and Gradients

strength



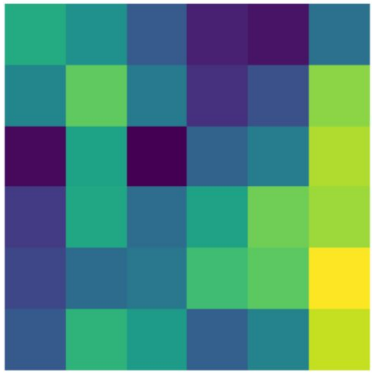
7x7 Feature Map

influence

7x7 Gradient Map

Activations and Gradients

How important is this map?

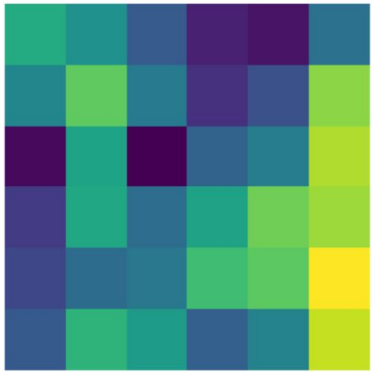


7x7 Feature Map

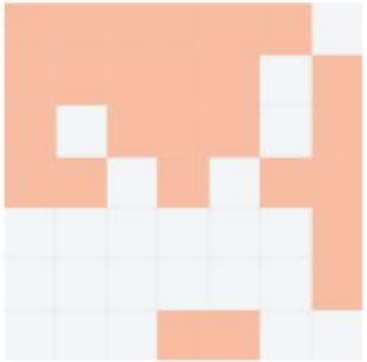
7x7 Gradient Map

Activations and Gradients

How important is this map?



7x7 Feature Map

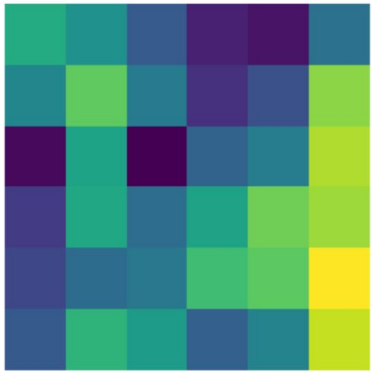


7x7 Gradient Map

```
weight = torch.mean(gradients, dim=(1, 2))
```

Activations and Gradients

How important is this map?



7x7 Feature Map

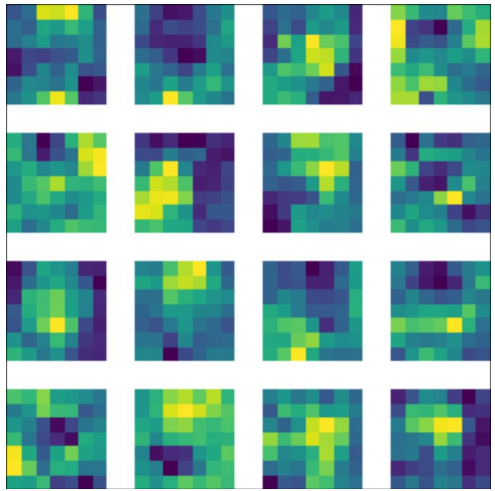
Average Influence

.004

```
weight = torch.mean(gradients, dim=(1, 2))
```

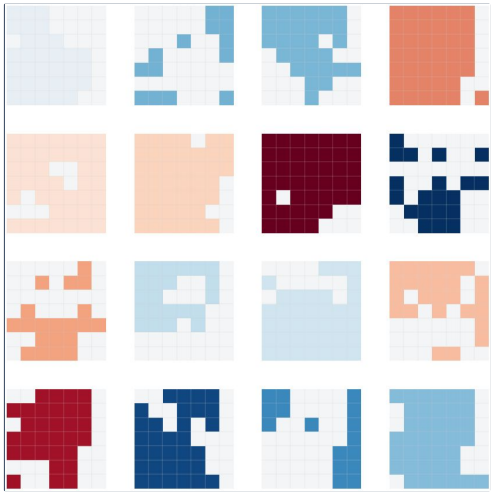
Activations and Gradients

Activations



2048 Feature Maps
7x7

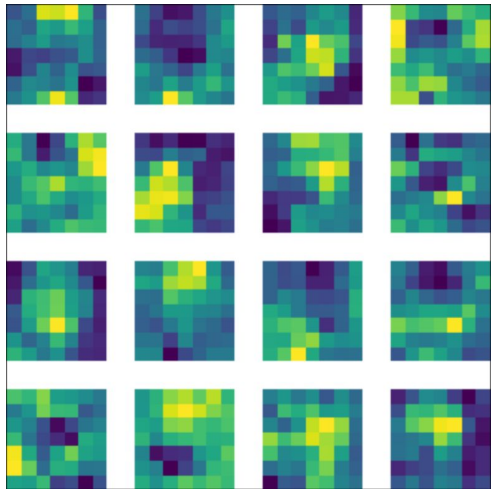
Gradients



2048 Gradient Maps
7x7

Activations and Gradients

Activations



2048 Feature Maps
7x7

Gradients

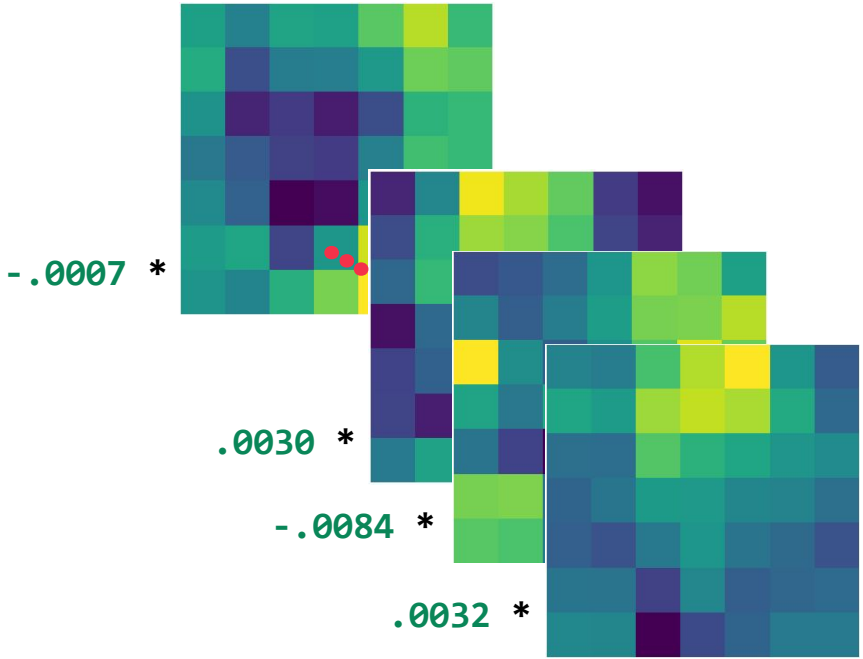
-.001	-.003	-.006	.010
.003	.005	.020	-.009
.004	-.002	-.003	.004
.013	-.013	-.005	-.007

2048 Gradient Maps
7x7

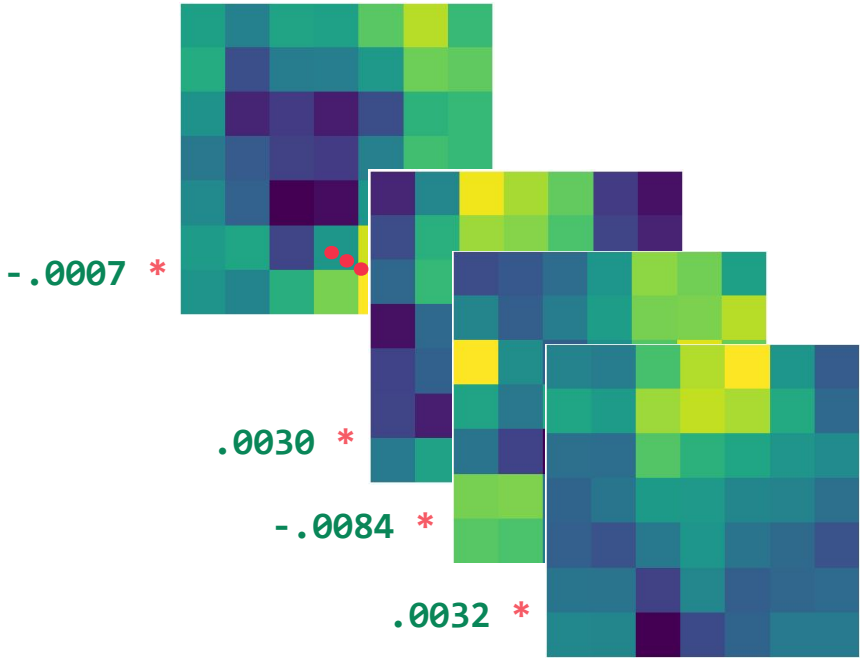
Weighted Maps

```
cam = torch.zeros(
    activations.shape[1:],
    dtype=activations.dtype).to(x.device)
```

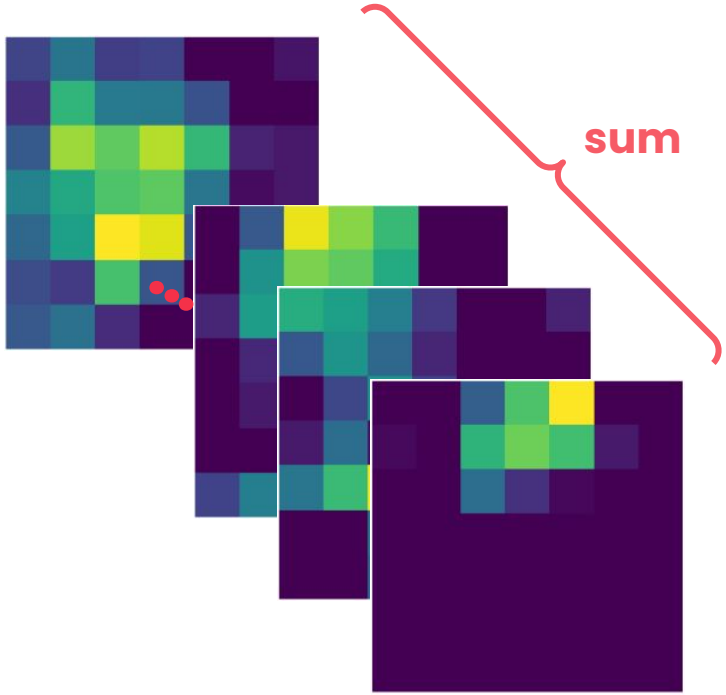
Weighted Maps



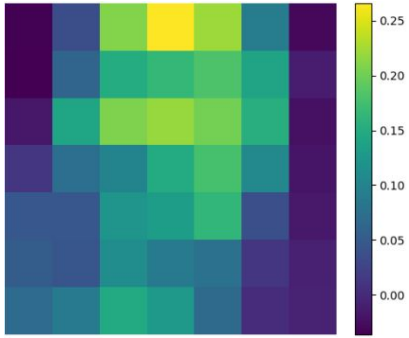
Weighted Maps



Weighted Maps

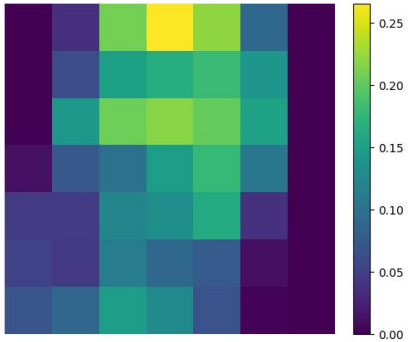


CAM Prep



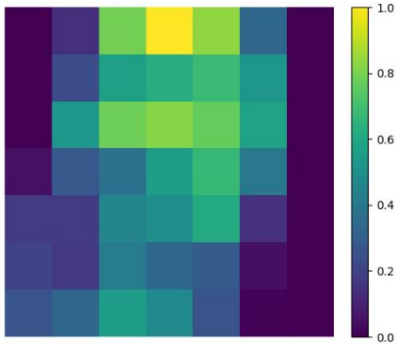

```
for i, w in enumerate(weights):
    cam += w * activations[i]
```

CAM Prep



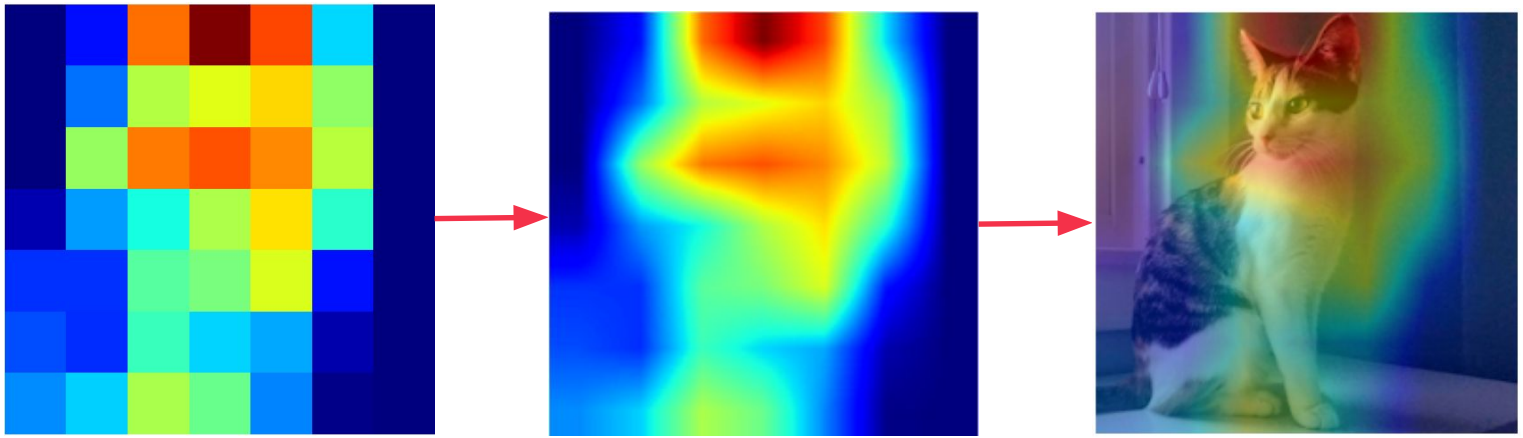

```
for i, w in enumerate(weights):
    cam += w * activations[i]
    # Keep positive values
    cam = torch.maximum(cam, torch.tensor(0, dtype=cam.dtype).to(x.device))
```

CAM Prep




```
for i, w in enumerate(weights):
    cam += w * activations[i]
    # Keep positive values
    cam = torch.maximum(cam, torch.tensor(0, dtype=cam.dtype).to(x.device))
    cam_np = cam.cpu().numpy()
    cam_np = (cam_np - cam_np.min()) / (cam_np.max() - cam_np.min() + 1e-8)
```

CAM Prep

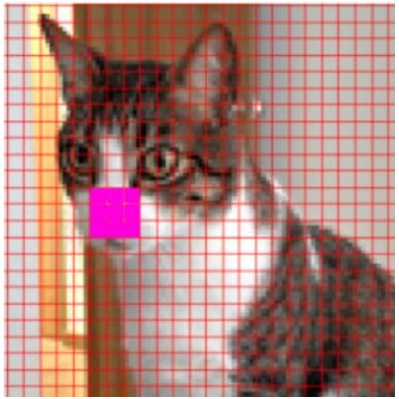


DeepLearning.AI

Diffusion

Specialized Approaches to Vision

Classifying and Interpreting Images



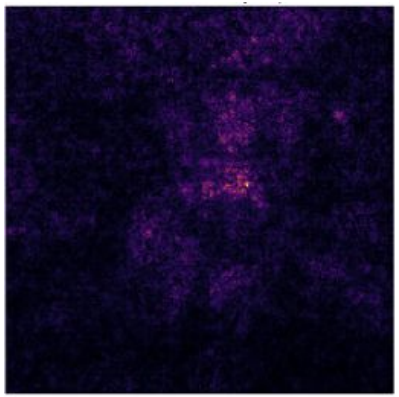


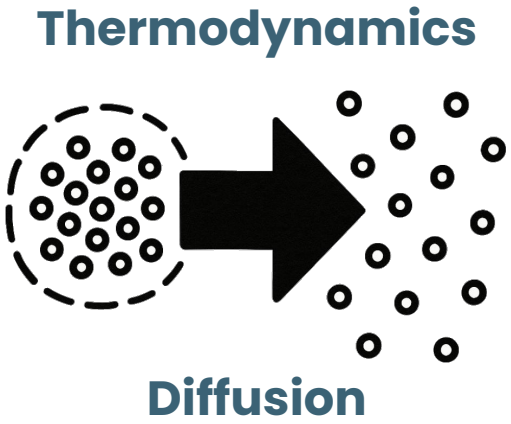
Image Generation

"A puppy riding a skateboard in Times Square"

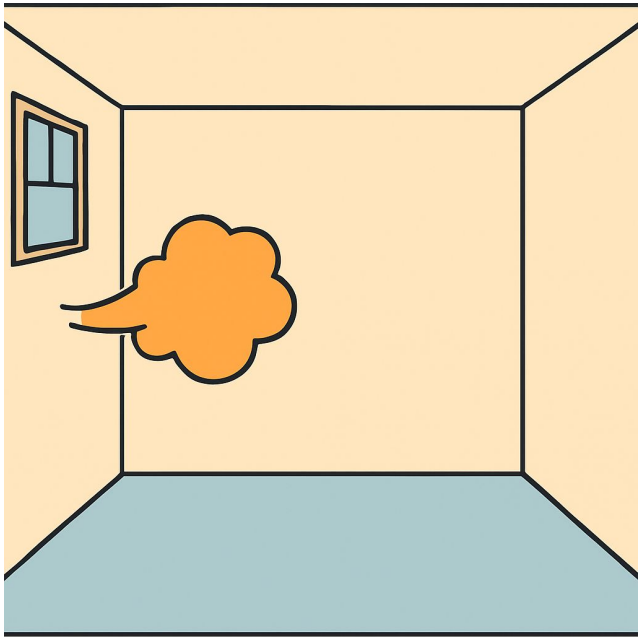
Early Image Generation

- **Produced blurry images.**
- **Unstable to train.**
- **Too slow for practical use.**

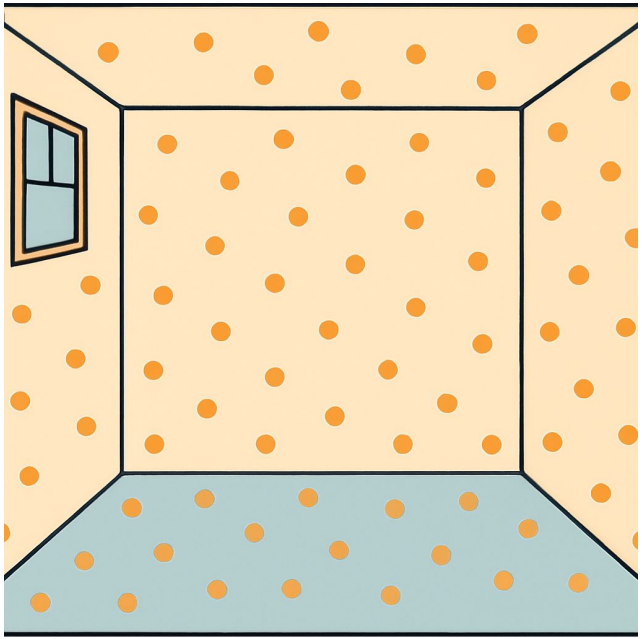
Modern Image Generation



Diffusion

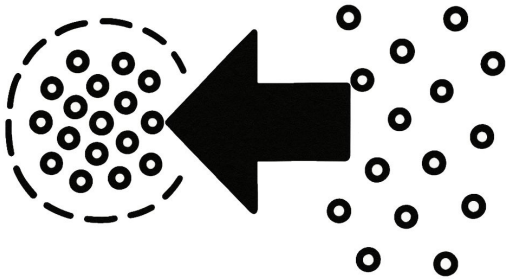


Diffusion



Follows probabilistic rules

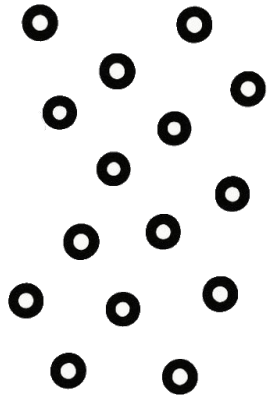
Reversing Diffusion



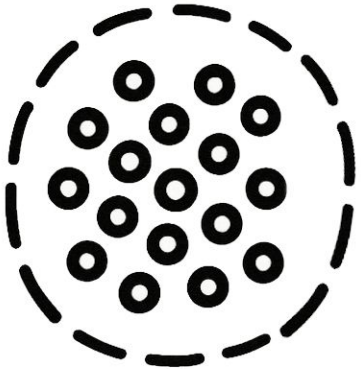
Probabilistic Rules

Reversing Diffusion

R: 102 G: 179 B: 92	R: 14 G: 106 B: 71	R: 188 G: 20 B: 102	R: 121 G: 210 B: 214	R: 74 G: 202 B: 87
R: 116 G: 99 B: 103	R: 151 G: 130 B: 149	R: 52 G: 1 B: 87	R: 235 G: 157 B: 37	R: 129 G: 191 B: 187
R: 20 G: 160 B: 203	R: 57 G: 21 B: 252	R: 235 G: 88 B: 48	R: 218 G: 58 B: 254	R: 169 G: 255 B: 219
R: 187 G: 207 B: 14	R: 189 G: 189 B: 174	R: 189 G: 50 B: 107	R: 54 G: 243 B: 63	R: 248 G: 130 B: 228
R: 50 G: 134 B: 20	R: 72 G: 166 B: 17	R: 131 G: 88 B: 59	R: 13 G: 241 B: 249	R: 8 G: 89 B: 52



Reversing Diffusion



Forward Diffusion

R:255 G:255 B:253	R:255 G:255 B:254	R:255 G:255 B:252	R:255 G:255 B:249	R:255 G:255 B:242
R:255 G:255 B:255	R:255 G:255 B:255	R:255 G:255 B:251	R:255 G:255 B:249	R:255 G:255 B:244
R:210 G:195 B:170	R:255 G:255 B:255	R:255 G:255 B:255	R:255 G:255 B:250	R:255 G:255 B:244
R: 87 G: 43 B: 11	R:164 G:146 B:121	R:255 G:255 B:252	R:255 G:255 B:255	R:255 G:255 B:247
R:136 G:104 B: 64	R: 85 G: 49 B: 10	R:143 G:119 B: 92	R:250 G:248 B:239	R:255 G:255 B:255

Latent Space

Forward Diffusion

Image Tensor

```
    R       G       B
[[ 0.45, -0.10, 0.72],
 [-0.30,  0.80, -.05],
 [ 0.00,  0.15, -0.60],
. . .]
```

+

Random Gaussian Noise Tensor

```
[[ 0.01, -0.05, 0.22],
 [-0.03,  0.02, -.09],
 [ 0.18, -0.07, -0.04],
. . .]
```

Forward Diffusion

Image + Noise

```
    R       G      B
[[ 0.46, -0.05, 0.94],

 [-0.27,  0.82, -.14],

 [ 0.18,  0.08, -0.64],

. . .]
```

Forward Diffusion

Image + Noise

```
    R        G        B
[[ 0.46, -0.05, 0.94],
 [-0.27,  0.82, -.14],
 [ 0.18,  0.08, -0.64],
. . .]
```

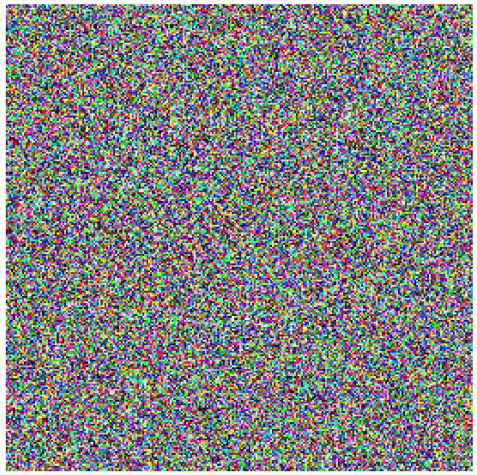
Random Gaussian Noise Tensor

```
[[ 0.01, -0.05, 0.22],
 [-0.03,  0.02, -.09],
 [ 0.18, -0.07, -0.04],
. . .]
```

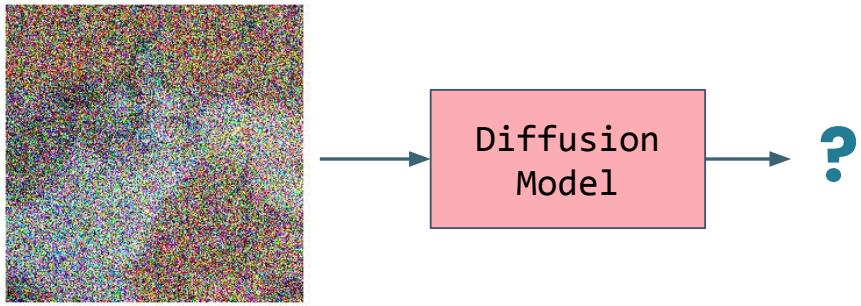
Forward Diffusion

Diffusion

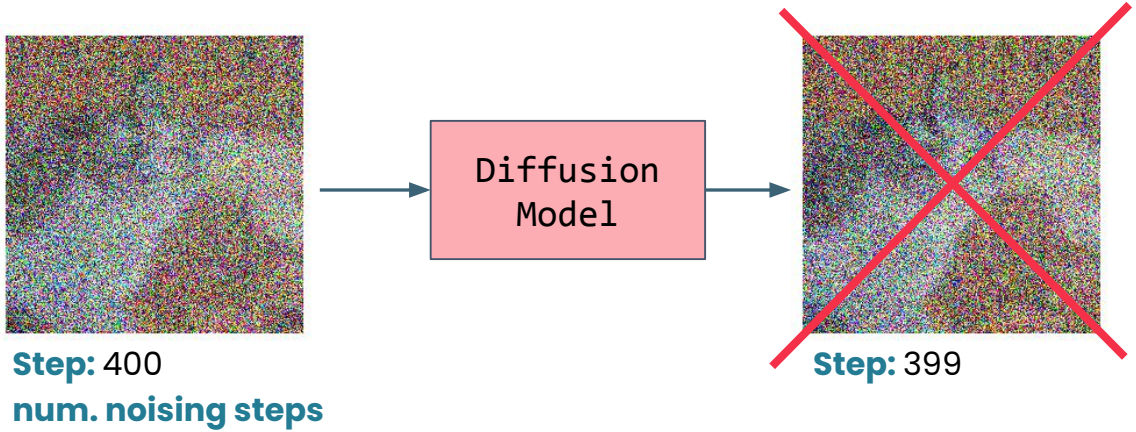
Diffusion



What does the model predict?



What does the model predict?



What does the model predict?

Step: 400

num. noising steps

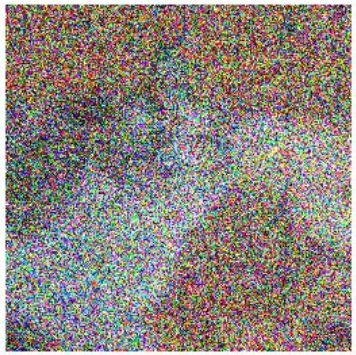
Diffusion Model

Random Gaussian Noise Tensor

```
[[ 0.71, -0.45, 0.22],
 [-0.93,  0.12, -.79],
 [ 0.18, -0.57, -0.44],
. . .]
```

Predicting Noise

Current Image
Step: 400

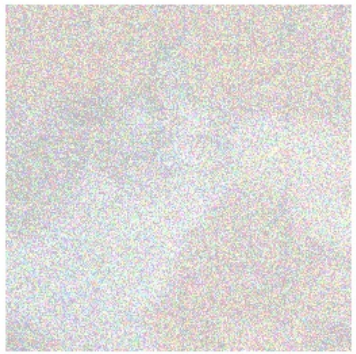


=

Earlier Image + **Noise Tensor**

Predicting Noise

Current Image
Step: 400



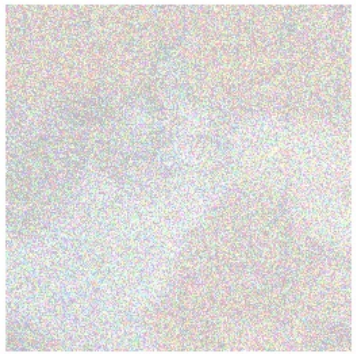
=

Model predicts:

Earlier Image + **Noise Tensor**

Predicting Noise

Current Image
Step: 400

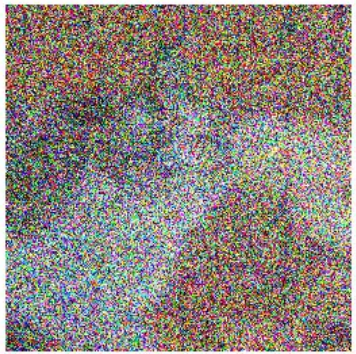


=

Earlier Image + **Noise Tensor**

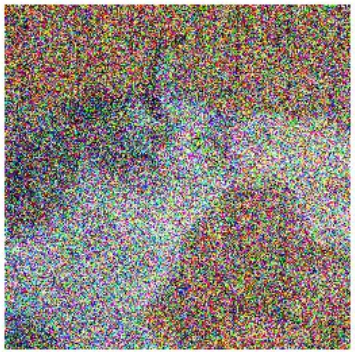
Predicting Noise

Current Image
Step: 400



=

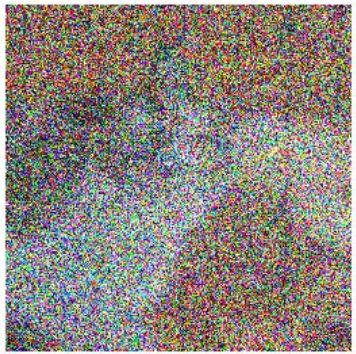
Earlier Image
Step: 399



+ **Noise Tensor**

Predicting Noise

Current Image
Step: 400



=

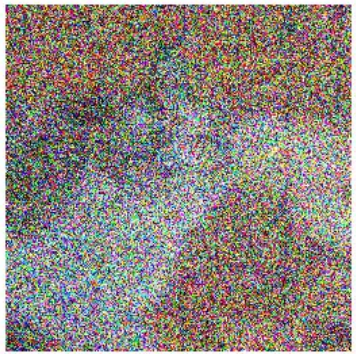
Earlier Image
Step: 399

Small noise from previous

```
[[ 0.01, -0.05, 0.22],
 [-0.03,  0.02, -.09],
  [ 0.18, -0.07, -0.04],
 . . .]
```

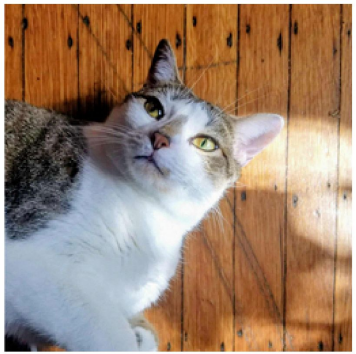
Predicting Noise

Current Image
Step: 400



=

From:
Original Image



Model predicts

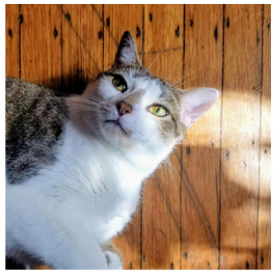
Total Accumulated Noise

```
[[ 1.71, -3.45, 2.22],
+ [-1.93,  2.12, -1.79],
  [ 4.18, -1.57, -3.44],
  . . .]
```

Training Data

Training Data

Original Image



+

Total Accumulated Noise

```
[[ 1.71, -3.45, 2.22],
 [-1.93,  2.12, -1.79],
 [ 4.18, -1.57, -3.44],
. . .]
```

=

Noisy image

t=125

Training Data

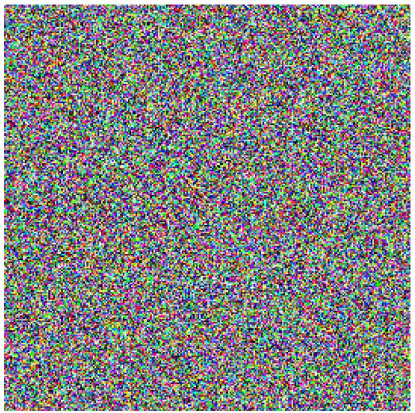
t=125

Diffusion Model

Total Accumulated Noise

```
[[ 1.71, -3.45, 2.22],
 [-1.93,  2.12, -1.79],
 [ 4.18, -1.57, -3.44],
. . .]
```

Total Accumulated Noise



−

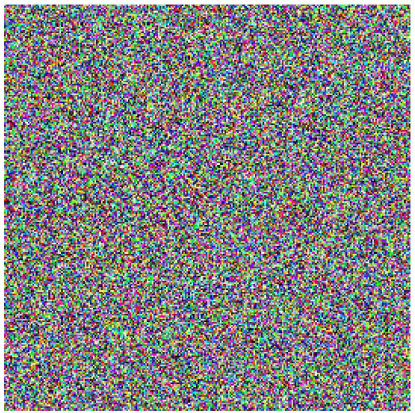
Total Accumulated Noise

```
[[ 1.71, -3.45, 2.22],
 [-1.93,  2.12, -1.79],
 [ 4.18, -1.57, -3.44],
. . .]
```

= **Clean image?**

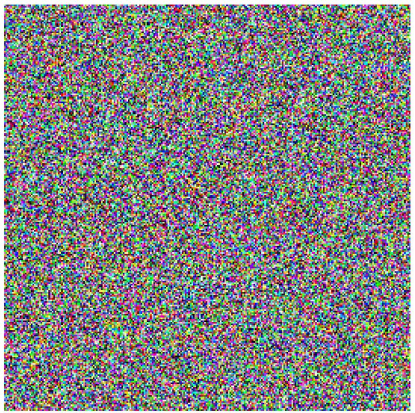
Image Generation

Why Predict Total Accumulated Noise?



− **Total Accumulated Noise** = ?

Removing Total Noise



750 noise steps

−

Total Accumulated Noise

=

Removing Total Noise

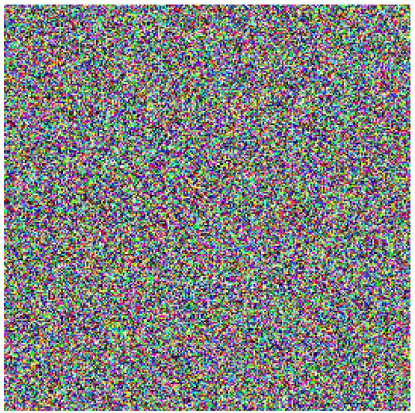
750 noise steps

−

Total Accumulated Noise

=

Gradual Generation



750 noise steps

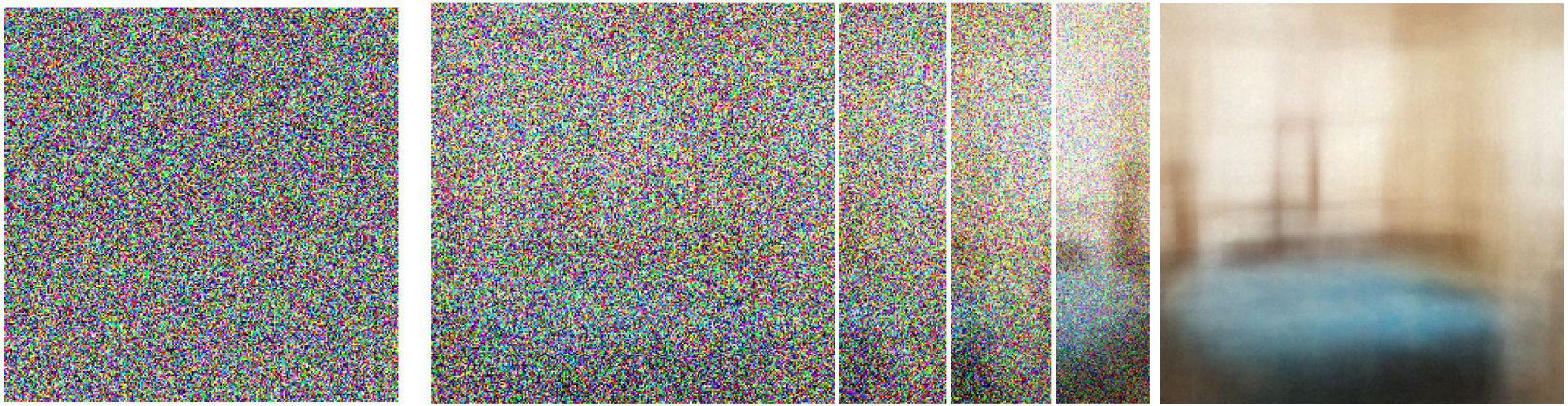
−

Total Accumulated Noise

=

Predicted Original

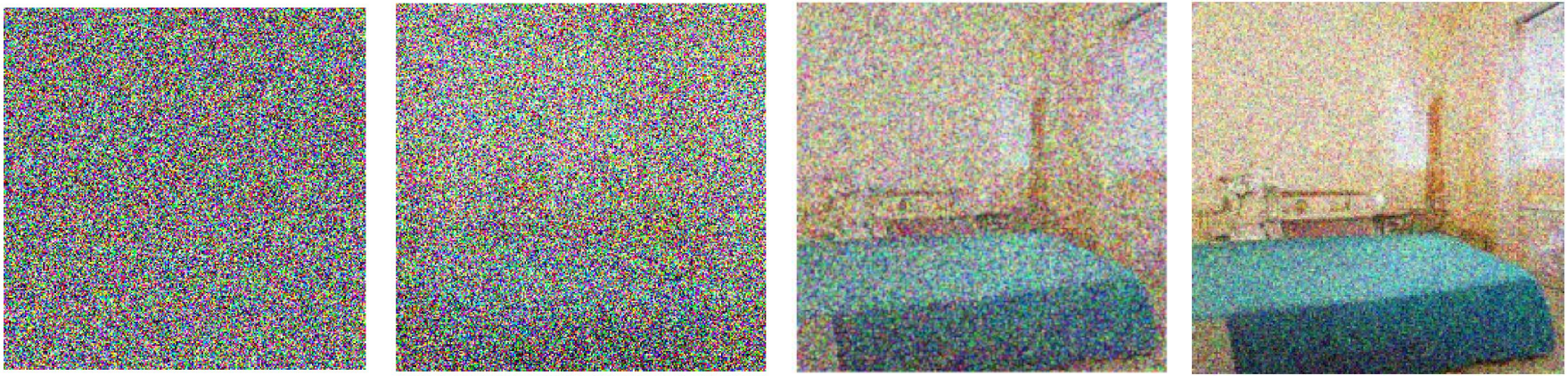
Gradual Generation



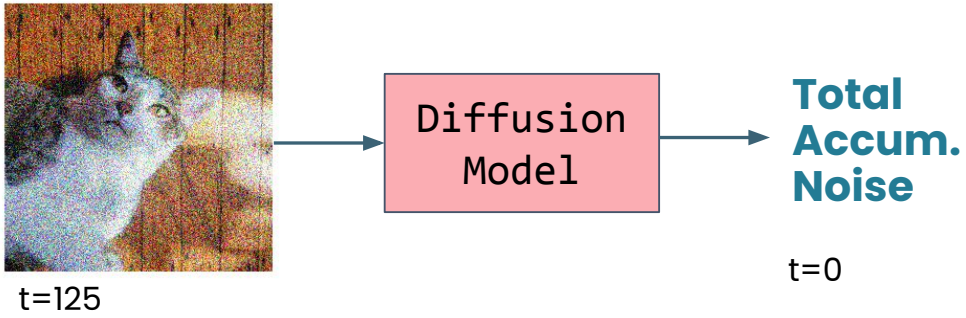
750 noise steps

749 noise steps

Gradual Generation



Why Predict Total Accumulated Noise?



Why Predict Total Accumulated Noise?

Predicting noise from noise

t=125

Diffusion Model

Previous Step Noise

t=124

Why Predict Total Accumulated Noise?

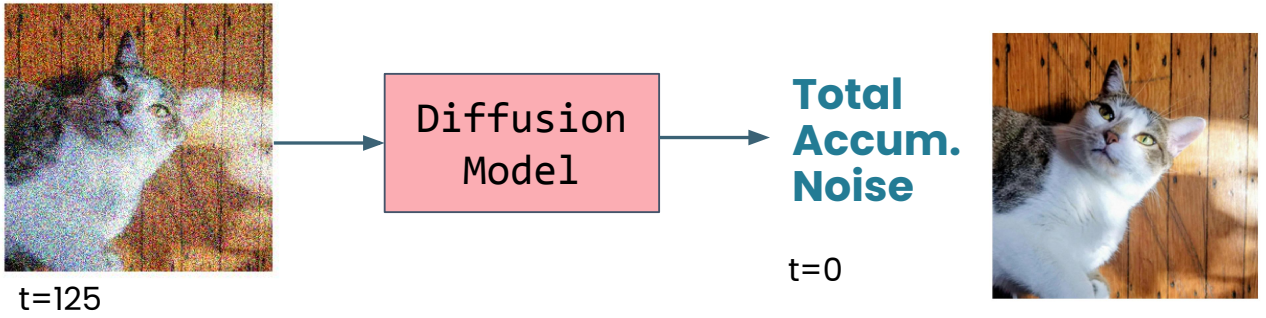
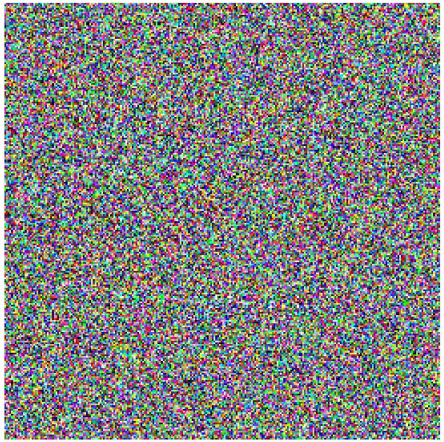


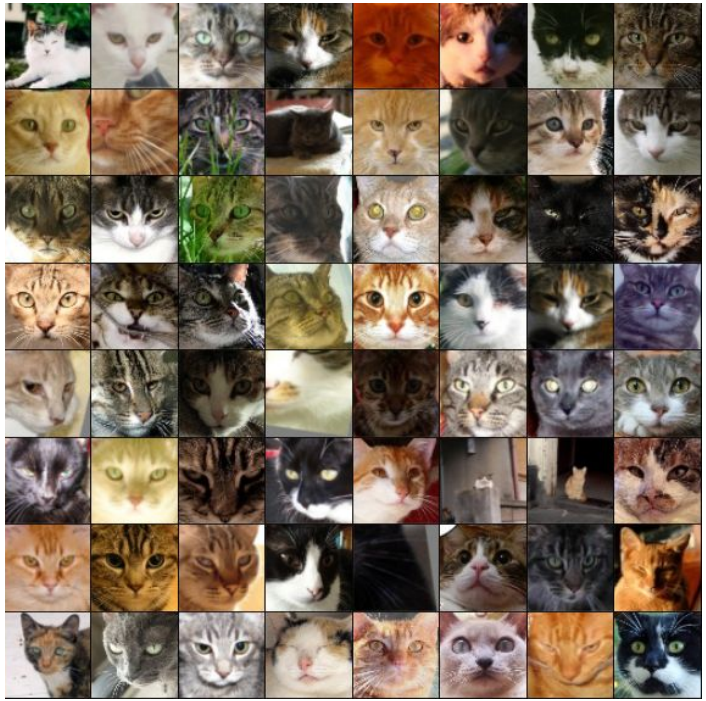
Image Generation

"A puppy riding a skateboard in Times Square"

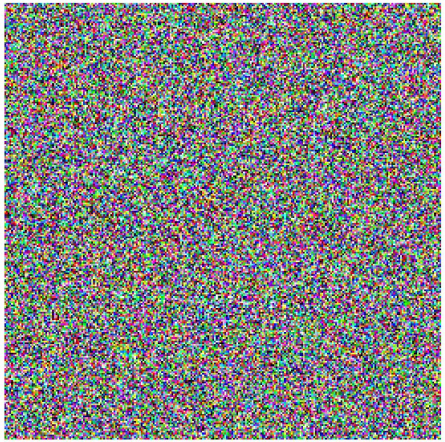
Unconditional Models



Unconditional Models

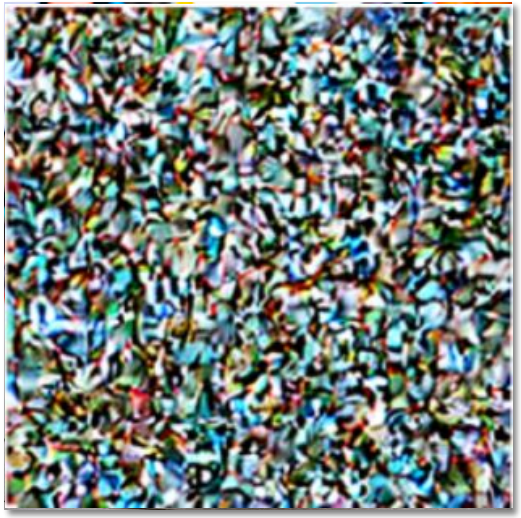


Cat Dataset © 2022 AnnikaV9, licensed under the MIT License



Conditioning

"A puppy riding a skateboard in Times Square"



Step: 400

DeepLearning.AI

Image Generation Walkthrough

Specialized Approaches to Vision

Diffusion

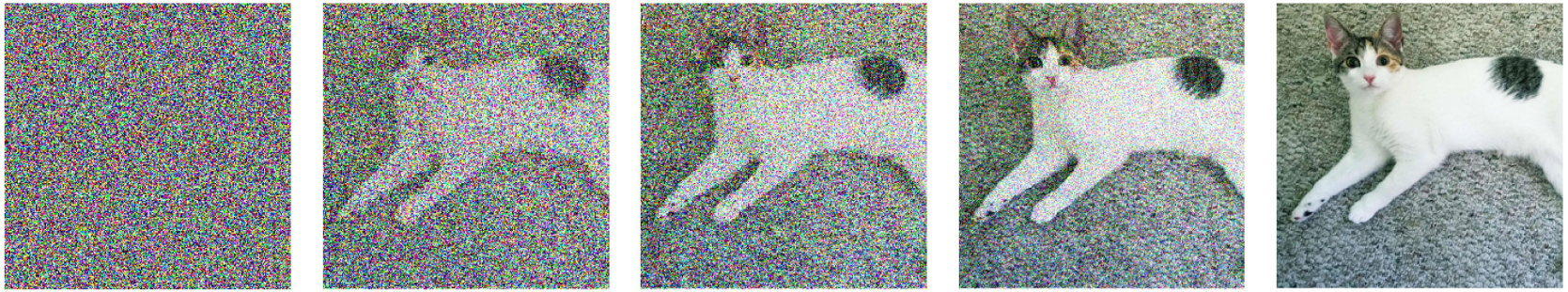


Image Generation

"A puppy riding a skateboard in Times Square"

Stable Diffusion

```
from diffusers import StableDiffusionPipeline
```

Stable Diffusion

```
from diffusers import StableDiffusionPipeline
```

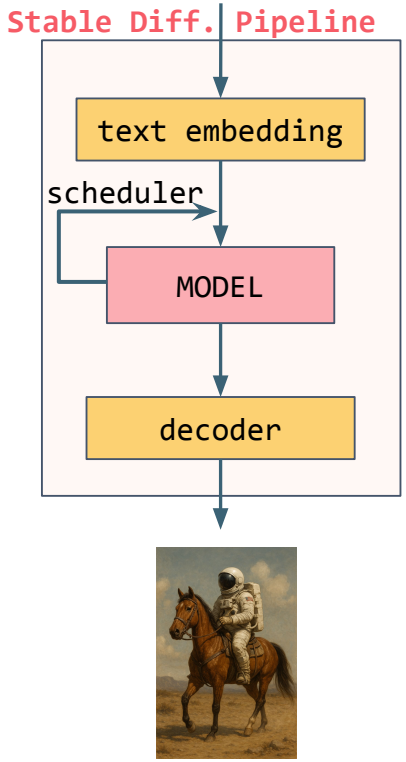
Stable Diffusion

```
from diffusers import StableDiffusionPipeline
```

Pipeline

```
from diffusers import StableDiffusionPipeline
```

“An astronaut riding a horse”



Pipeline

```
from diffusers import StableDiffusionPipeline
```

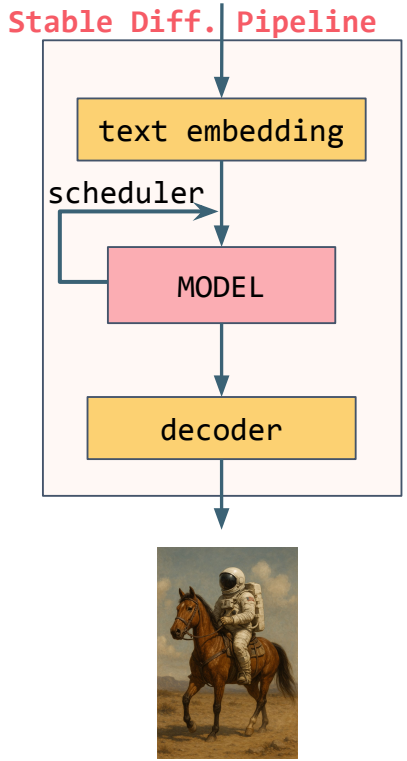
Diffusers Library

```
from diffusers import StableDiffusionPipeline

repo_id = "stabilityai/stable-diffusion-2-base"


          "stable-diffusion-3.5-medium"
          "stable-diffusion-3.5-large"
          "stable-diffusion-3.5-large-turbo"
```

Diffusers Library

```
from diffusers import StableDiffusionPipeline

repo_id = "stabilityai/stable-diffusion-2-base"

pipe = StableDiffusionPipeline.from_pretrained(repo_id,
                                               torch_dtype=torch.float16
                                               variant="fp16",
                                               cache_dir="./.models")
```

"An astronaut riding a horse"



Diffusers Library

```
from diffusers import StableDiffusionPipeline

repo_id = "stabilityai/stable-diffusion-2-base"

pipe = StableDiffusionPipeline.from_pretrained(repo_id,
                                               torch_dtype=torch.float16
                                               variant="fp16",
                                               cache_dir="./.models")
```

Diffusers Library

```
from diffusers import StableDiffusionPipeline

repo_id = "stabilityai/stable-diffusion-2-base"

pipe = StableDiffusionPipeline.from_pretrained(repo_id,
                                               torch_dtype=torch.float16
                                               variant="fp16",
                                               cache_dir="./.models")
```

Diffusers Library

```
from diffusers import StableDiffusionPipeline

repo_id = "stabilityai/stable-diffusion-2-base"

pipe = StableDiffusionPipeline.from_pretrained(repo_id,
                                               torch_dtype=torch.float16
                                               variant="fp16",
                                               cache_dir="./.models")
```

Diffusers Library

```
from diffusers import StableDiffusionPipeline

repo_id = "stabilityai/stable-diffusion-2-base"

pipe = StableDiffusionPipeline.from_pretrained(repo_id,
                                               torch_dtype=torch.float16
                                               variant="fp16",
                                               cache_dir="./.models")
```

Diffusers Library

```
from diffusers import StableDiffusionPipeline

repo_id = "stabilityai/stable-diffusion-2-base"

pipe = StableDiffusionPipeline.from_pretrained(repo_id,
                                               torch_dtype=torch.float16
                                               variant="fp16",
                                               cache_dir="./.models")
```

Diffusers Library

```
from diffusers import StableDiffusionPipeline

repo_id = "stabilityai/stable-diffusion-2-base"

pipe = StableDiffusionPipeline.from_pretrained(repo_id, . . .)

generator = torch.Generator(device=device).manual_seed(42)
```

Diffusers Library

```
from diffusers import StableDiffusionPipeline

repo_id = "stabilityai/stable-diffusion-2-base"

pipe = StableDiffusionPipeline.from_pretrained(repo_id, . . .)

generator = torch.Generator(device=device).manual_seed(42)
```

Diffusers Library

```
from diffusers import StableDiffusionPipeline

repo_id = "stabilityai/stable-diffusion-2-base"

pipe = StableDiffusionPipeline.from_pretrained(repo_id, . . .)

generator = torch.Generator(device=device).manual_seed(42)
pipe = pipe.to(device)
prompt = "A puppy riding a skateboard in Times Square."

images = pipe(
    prompt,
    num_inference_steps=40,
    generator=generator
).images
```

Diffusers Library

```
from diffusers import StableDiffusionPipeline

repo_id = "stabilityai/stable-diffusion-2-base"

pipe = StableDiffusionPipeline.from_pretrained(repo_id, . . .)

generator = torch.Generator(device=device).manual_seed(42)
pipe = pipe.to(device)
prompt = "A puppy riding a skateboard in Times Square."

images = pipe(
    prompt,
    num_inference_steps=40,
    generator=generator
).images
```

Diffusers Library

```
from diffusers import StableDiffusionPipeline

repo_id = "stabilityai/stable-diffusion-2-base"

pipe = StableDiffusionPipeline.from_pretrained(repo_id, . . .)

generator = torch.Generator(device=device).manual_seed(42)
pipe = pipe.to(device)
prompt = "A puppy riding a skateboard in Times Square."

images = pipe(
    prompt,
    num_inference_steps=40,
    generator=generator
).images
```

Diffusers Library

```
from diffusers import StableDiffusionPipeline

repo_id = "stabilityai/stable-diffusion-2-base"

pipe = StableDiffusionPipeline.from_pretrained(repo_id, . . .)

generator = torch.Generator(device=device).manual_seed(42)
pipe = pipe.to(device)
prompt = "A puppy riding a skateboard in Times Square."

images = pipe(
    prompt,
    num_inference_steps=40,
    generator=generator
).images
```

more steps = more refinement

Diffusers Library

```
from diffusers import StableDiffusionPipeline

repo_id = "stabilityai/stable-diffusion-2-base"

pipe = StableDiffusionPipeline.from_pretrained(repo_id, . . .)

generator = torch.Generator(device=device).manual_seed(42)
pipe = pipe.to(device)
prompt = "A puppy riding a skateboard in Times Square."

images = pipe(
    prompt,
    num_inference_steps=40,
    generator=generator
).images
```

Stable Diffusion

```
generator = torch.Generator(device=device).manual_seed(42)
```

Parameters

```
image_list = pipe(
    prompt,
    num_images_per_prompt=num_im,
    num_inference_steps=num_inf,
    generator=generator,
    guidance_scale=gs,
    negative_prompt=neg_prompt
).images
```

Parameters

```
image_list = pipe(
    prompt,
    num_images_per_prompt=num_im,
    num_inference_steps=num_inf,
    generator=generator,
    guidance_scale=gs,
    negative_prompt=neg_prompt
).images
```

Parameters

```
image_list = pipe(
    prompt,
    num_images_per_prompt=3,
    num_inference_steps=num_inf,
    generator=generator,
    guidance_scale=gs,
    negative_prompt=neg_prompt
).images
```

Parameters

```
image_list = pipe(
    prompt,
    num_images_per_prompt=num_im
    num_inference_steps=num_inf,
    generator=generator,
    guidance_scale=gs,
    negative_prompt=neg_prompt
).images
```

```
prompt = "A cute dog with a red bandana, sitting in a lush park"
```

Parameters

```
image_list = pipe(
    prompt,
    num_images_per_prompt=num_im
    num_inference_steps=num_inf,
    generator=generator,
    guidance_scale=gs,
    negative_prompt=neg_prompt
).images
```

```
prompt = "A surreal landscape with floating clocks, melting
trees, and a purple sky, in the style of Salvador Dali"
```

guidance_scale = 5

Parameters

```
image_list = pipe(
    prompt,
    num_images_per_prompt=num_im
    num_inference_steps=num_inf,
    generator=generator,
    guidance_scale=gs,
    negative_prompt=neg_prompt
).images
```

```
neg_prompt = "deformed hands"
```

Parameters

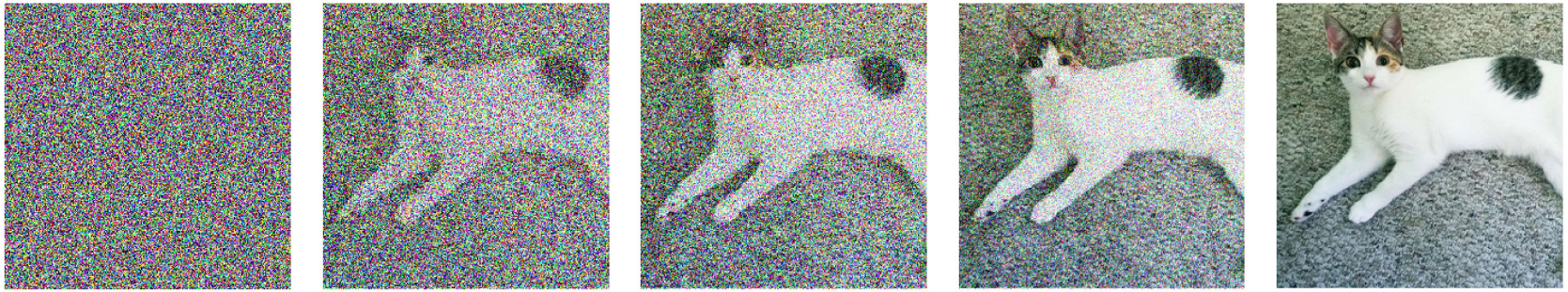
```
image_list = pipe(
    prompt,
    num_images_per_prompt=num_im
    num_inference_steps=num_inf,
    generator=generator,
    guidance_scale=gs,
    negative_prompt=neg_prompt
).images
```

```
prompt = "Teddy Bears eating pizza at a birthday party”
neg_prompt = "no pepperoni pizza"
```

Parameters

```
image_list = pipe(
    prompt,
    num_images_per_prompt=num_im,
    num_inference_steps=num_inf,
    generator=generator,
    guidance_scale=gs
    negative_prompt=neg_prompt
).images
```

Visualizing Denoising



Latent Space

Latent Space

- **Smaller**
- **Faster and Uses Less Memory**
- **Scalable and Practical**
- **Check out the resources!**