

## 02\_ML-Durchlauf\_gesamt

June 2, 2021

```
[1]: from fastai.data.external import *  
     from fastai.vision.all import *
```

```
[2]: path = untar_data(URLs.DOGS)  
     path.ls()
```

```
[2]: (#6) [Path('/home/jovyan/.fastai/data/dogscats/test1'),Path('/home/jovyan/.fastai/data/dogscats/config.yml'),Path('/home/jovyan/.fastai/data/dogscats/archive'),Path('/home/jovyan/.fastai/data/dogscats/sample'),Path('/home/jovyan/.fastai/data/dogscats/train'),Path('/home/jovyan/.fastai/data/dogscats/valid')]
```

```
[3]: files = get_image_files(path / "train")  
     len(files)
```

```
[3]: 23000
```

```
[4]: files[0], files[20000]
```

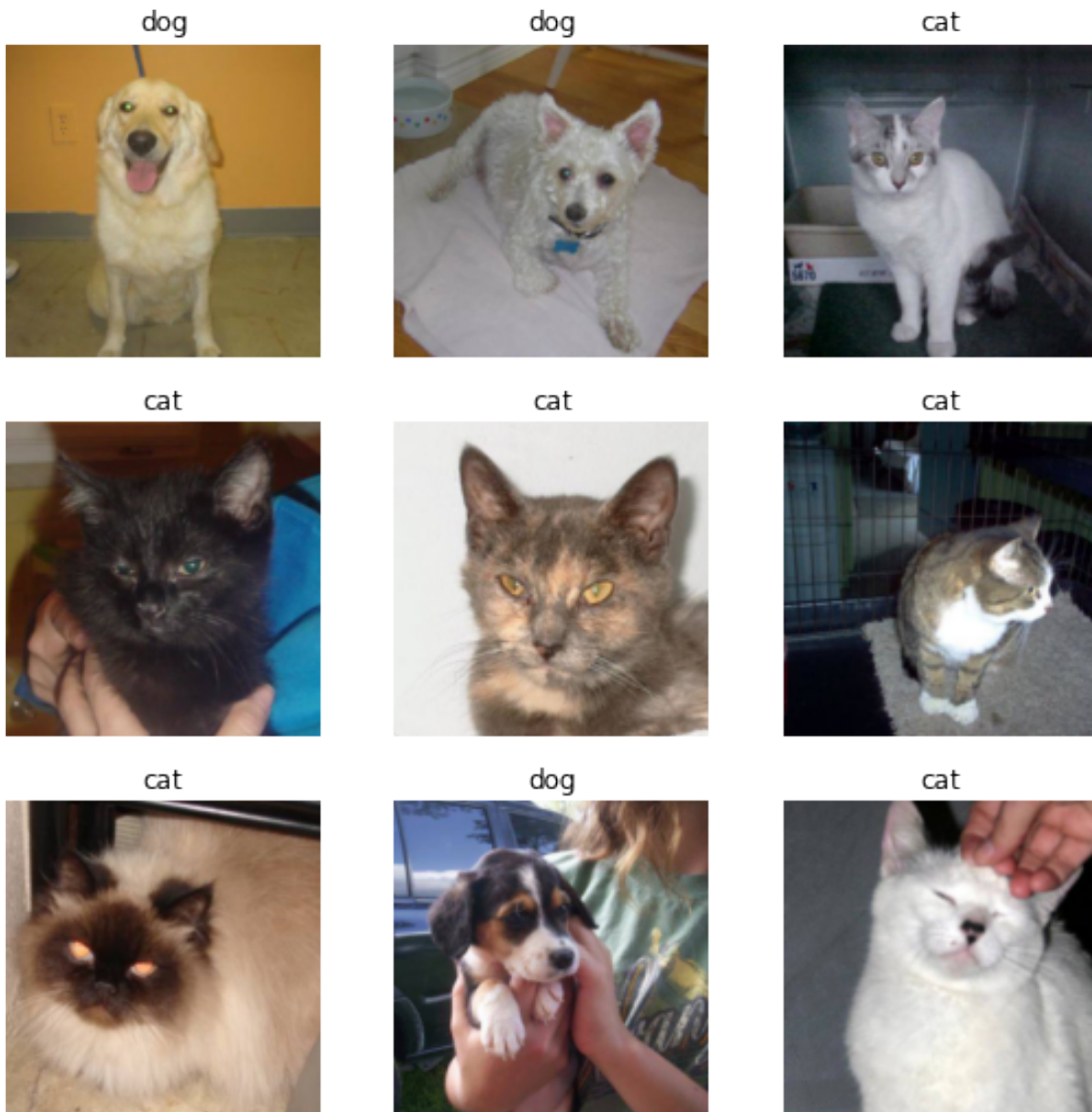
```
[4]: (Path('/home/jovyan/.fastai/data/dogscats/train/dogs/dog.3410.jpg'),  
     Path('/home/jovyan/.fastai/data/dogscats/train/cats/cat.9413.jpg'))
```

```
[5]: def label_func(f):  
     return "cat" if f.startswith("cat") else "dog"
```

```
[6]: dls = ImageDataLoaders.from_name_func(  
     path / "train", files, label_func, item_tfms=Resize(224)  
     )
```

```
[7]: print(dls.n)  
     dls.show_batch()
```

```
18400
```



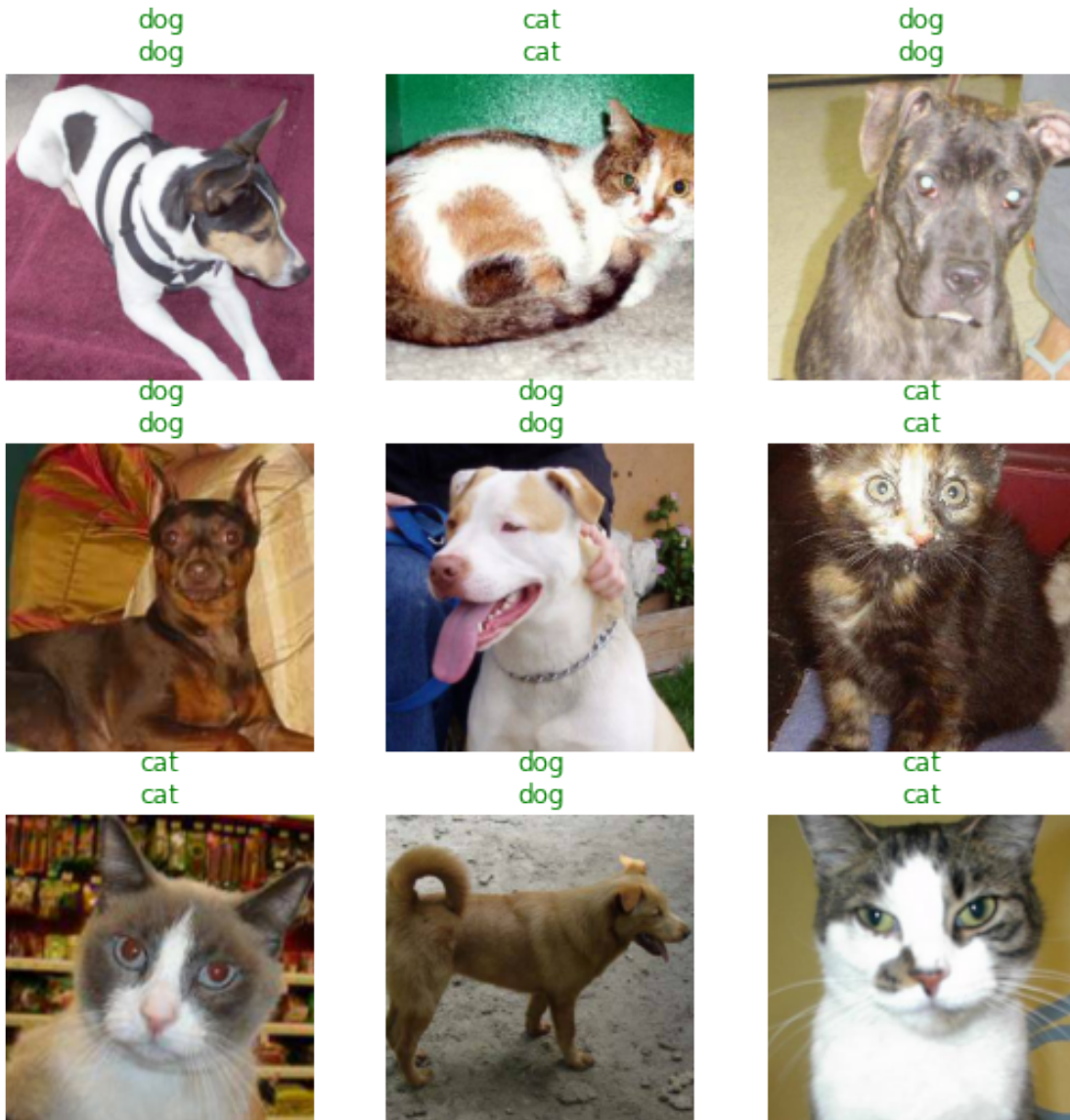
```
[8]: learn = cnn_learner(dls, resnet34, metrics=error_rate)
      learn.fine_tune(1)
```

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

```
[9]: learn.show_results()
```

<IPython.core.display.HTML object>



```
[10]: path.ls()
```

```
[10]: (#6) [Path('/home/jovyan/.fastai/data/dogscats/test1'),Path('/home/jovyan/.fastai/data/dogscats/config.yml'),Path('/home/jovyan/.fastai/data/dogscats/archive'),Path('/home/jovyan/.fastai/data/dogscats/sample'),Path('/home/jovyan/.fastai/data/dogscats/train'),Path('/home/jovyan/.fastai/data/dogscats/valid')]
```

```
[11]: test_files = get_image_files(path / "valid")
test_files[0]
```

```
[11]: Path('/home/jovyan/.fastai/data/dogscats/valid/dogs/dog.5728.jpg')
```

```
[12]: # is it a dog or a cat?
learn.predict(test_files[0])
```

<IPython.core.display.HTML object>

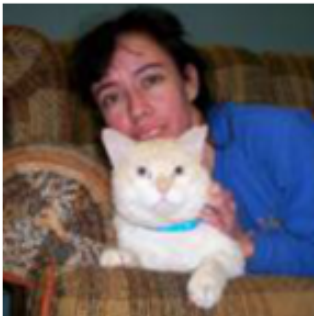
```
[12]: ('dog', tensor(1), tensor([7.9356e-06, 9.9999e-01]))
```

```
[13]: interp = Interpretation.from_learner(learn)
interp.plot_top_losses(9)
```

<IPython.core.display.HTML object>

### Prediction/Actual/Loss/Probability

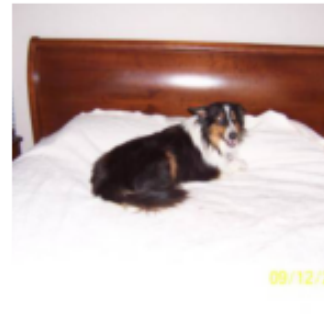
cat/dog / 9.58 / 1.00



cat/dog / 5.03 / 0.99



cat/dog / 4.75 / 0.99



cat/dog / 4.37 / 0.99



dog/cat / 4.33 / 0.99



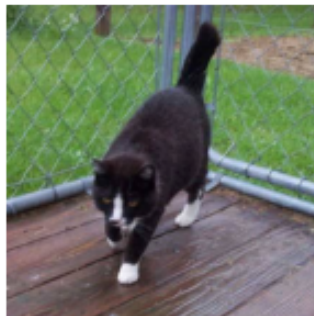
cat/dog / 4.15 / 0.98



cat/dog / 3.49 / 0.97



dog/cat / 3.25 / 0.96



dog/cat / 2.93 / 0.95



## 0.1 Free GPU Memory

```
[14]: def reload(learner):  
      import gc  
  
      model, opt = learner.model, learner.opt  
      test_dls = ImageDataLoaders.from_name_func(  
          path / "test", files, label_func, item_tfms=Resize(224), bs=1  
      )  
  
      del learner  
      gc.collect()  
      torch.cuda.empty_cache()  
      return Learner(test_dls, model)
```

```
[15]: learner = reload(learn)  
      learner
```

```
[15]: <fastai.learner.Learner at 0x7f8734eec040>
```

## 0.2 Explaining using Lime

```
[16]: !pip install lime
```

```
Requirement already satisfied: lime in /opt/conda/lib/python3.9/site-packages  
(0.2.0.1)  
Requirement already satisfied: scikit-learn>=0.18 in  
/opt/conda/lib/python3.9/site-packages (from lime) (0.24.2)  
Requirement already satisfied: matplotlib in /opt/conda/lib/python3.9/site-  
packages (from lime) (3.4.2)  
Requirement already satisfied: scikit-image>=0.12 in  
/opt/conda/lib/python3.9/site-packages (from lime) (0.18.1)  
Requirement already satisfied: scipy in /opt/conda/lib/python3.9/site-packages  
(from lime) (1.6.3)  
Requirement already satisfied: tqdm in /opt/conda/lib/python3.9/site-packages  
(from lime) (4.61.0)  
Requirement already satisfied: numpy in /opt/conda/lib/python3.9/site-packages  
(from lime) (1.20.3)  
Requirement already satisfied: tifffile>=2019.7.26 in  
/opt/conda/lib/python3.9/site-packages (from scikit-image>=0.12->lime)  
(2021.4.8)  
Requirement already satisfied: PyWavelets>=1.1.1 in  
/opt/conda/lib/python3.9/site-packages (from scikit-image>=0.12->lime) (1.1.1)  
Requirement already satisfied: networkx>=2.0 in /opt/conda/lib/python3.9/site-
```



```

packages (from scikit-image>=0.12->lime) (2.5.1)
Requirement already satisfied: pillow!=7.1.0,!7.1.1,>=4.3.0 in
/opt/conda/lib/python3.9/site-packages (from scikit-image>=0.12->lime) (8.2.0)
Requirement already satisfied: imageio>=2.3.0 in /opt/conda/lib/python3.9/site-
packages (from scikit-image>=0.12->lime) (2.9.0)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.9/site-
packages (from matplotlib->lime) (0.10.0)
Requirement already satisfied: python-dateutil>=2.7 in
/opt/conda/lib/python3.9/site-packages (from matplotlib->lime) (2.8.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/opt/conda/lib/python3.9/site-packages (from matplotlib->lime) (1.3.1)
Requirement already satisfied: pyparsing>=2.2.1 in
/opt/conda/lib/python3.9/site-packages (from matplotlib->lime) (2.4.7)
Requirement already satisfied: six in /opt/conda/lib/python3.9/site-packages
(from cycler>=0.10->matplotlib->lime) (1.16.0)
Requirement already satisfied: decorator<5,>=4.3 in
/opt/conda/lib/python3.9/site-packages (from networkx>=2.0->scikit-
image>=0.12->lime) (4.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/opt/conda/lib/python3.9/site-packages (from scikit-learn>=0.18->lime) (2.1.0)
Requirement already satisfied: joblib>=0.11 in /opt/conda/lib/python3.9/site-
packages (from scikit-learn>=0.18->lime) (1.0.1)

```

```

[17]: def batch_predict(images):
        import torch
        from torchvision import transforms

        normalize = transforms.Normalize(
            mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]
        )
        transf = transforms.Compose([transforms.ToTensor(), normalize])
        model = learner.model
        model.eval()
        batch = torch.stack(tuple(transf(i) for i in images), dim=0)

        device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
        model.to(device)
        batch = batch.to(device)

        logits = model(batch)
        probs = F.softmax(logits, dim=1)
        return probs.detach().cpu().numpy()

```

```

[18]: from lime import lime_image
        from skimage.segmentation import mark_boundaries

```

```

def explain_image(dls_image_item):
    image_to_explain = to_image(dls_image_item[0])
    explainer = lime_image.LimeImageExplainer()
    explanation = explainer.explain_instance(
        np.array(image_to_explain),
        classifier_fn=batch_predict,
        labels=("cat", "dog"),
        hide_color=0,
    )
    temp, mask = explanation.get_image_and_mask(
        explanation.top_labels[0],
        positive_only=False,
        num_features=2,
        hide_rest=False,
    )
    description = explanation.top_labels[0]
    return mark_boundaries(temp / 255.0, mask), description

```

Explain random picks

```

[19]: import random

import matplotlib.pyplot as plt

RANDOM_COUNT = 12

selection = [random.randint(0, dls.n - 1) for _ in range(RANDOM_COUNT)]
items = [dls.do_item(item) for item in selection]

explained = []
descriptions = []
for item in items:
    explaining_image, description = explain_image(item)
    explained.append(explaining_image)
    descriptions.append(description)

```

```
0%|          | 0/1000 [00:00<?, ?it/s]
```

```
0%|          | 0/1000 [00:00<?, ?it/s]
```

```
0%|          | 0/1000 [00:00<?, ?it/s]
```

```
0%|          | 0/1000 [00:00<?, ?it/s]
```

```
0%|          | 0/1000 [00:00<?, ?it/s]
```

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

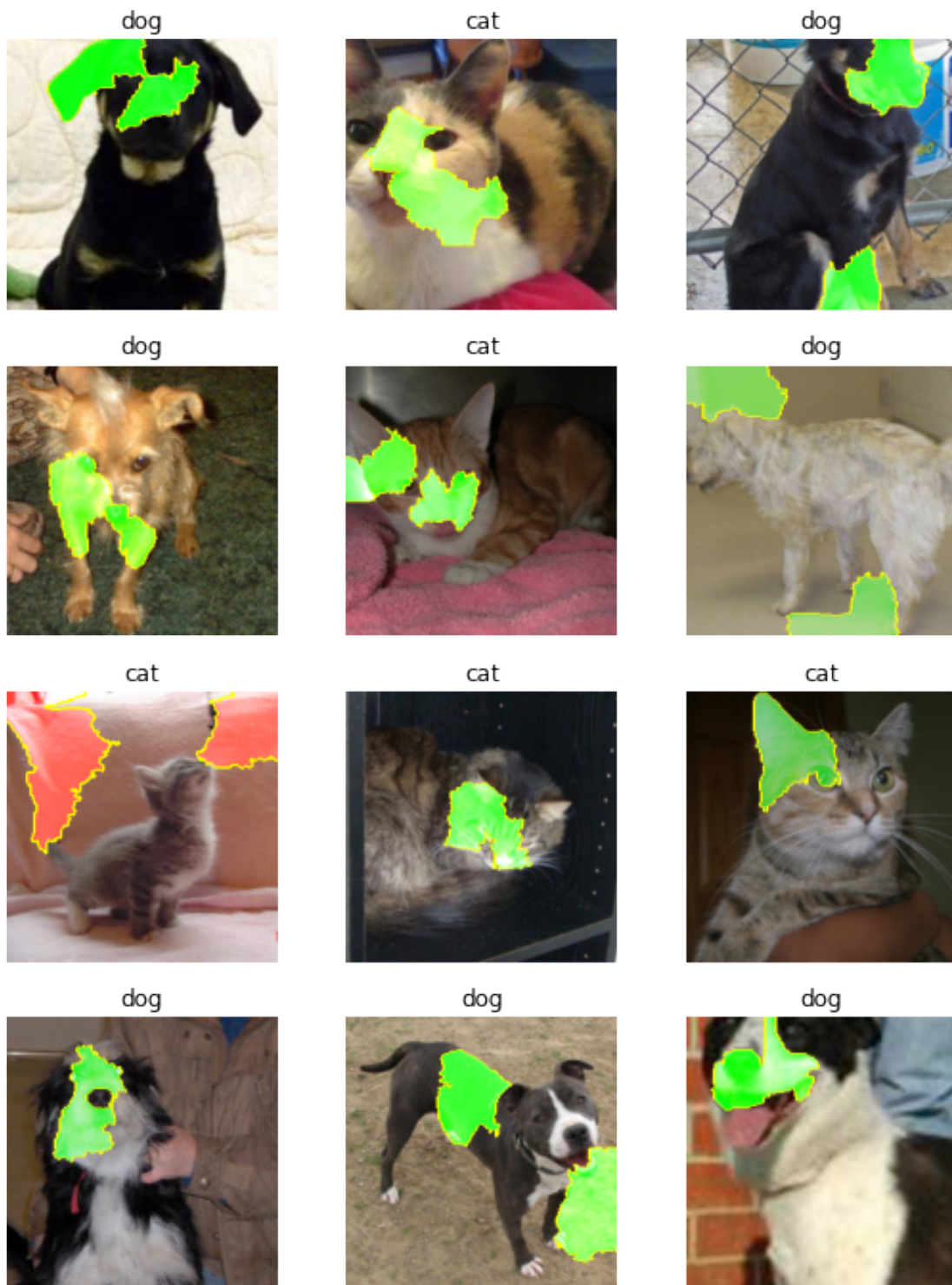
0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

```
[20]: labels = [dls.vocab[description] for description in descriptions]
      show_image_batch((explained, labels), items=len(descriptions))
```





```
[21]: explained = []
      descriptions = []
```

```
for item in items:
    explaining_image, description = explain_image(item)
    explained.append(explaining_image)
    descriptions.append(description)
show_image_batch((explained, labels), items=len(descriptions))
```

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]



```
[22]: explained = []
      descriptions = []
```

```
for item in items:
    explaining_image, description = explain_image(item)
    explained.append(explaining_image)
    descriptions.append(description)
show_image_batch((explained, labels), items=len(descriptions))
```

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]



```
[23]: explained = []
      descriptions = []
```



```

for item in items:
    explaining_image, description = explain_image(item)
    explained.append(explaining_image)
    descriptions.append(description)
show_image_batch((explained, labels), items=len(descriptions))

```

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]

0%| | 0/1000 [00:00<?, ?it/s]



### 0.3 Interpret Model

Using [captum](#) as an example library.



## Algorithm Description

```
[24]: learner = reload(learner)

from random import randint

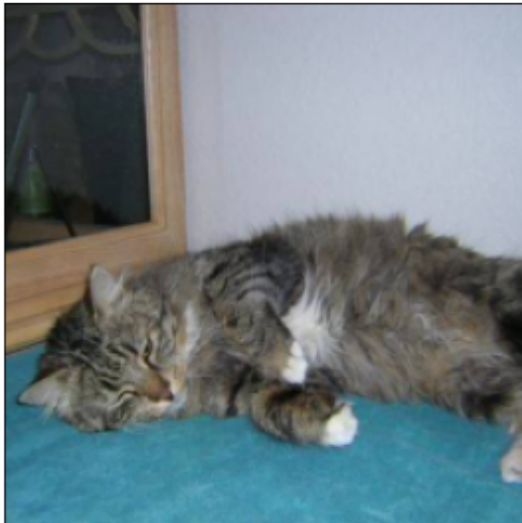
cat_index = 20089
dog_index = 44

# see above for what files are
# files = get_image_files(path / "train")
# idx = randint(0, len(files))
# image_to_interpret = files[idx]
cat_example = files[cat_index]
dog_example = files[dog_index]
img_cat = Image.open(cat_example).to_thumb(256)
img_dog = Image.open(dog_example).to_thumb(256)
display(img_cat), display(img_dog);
```

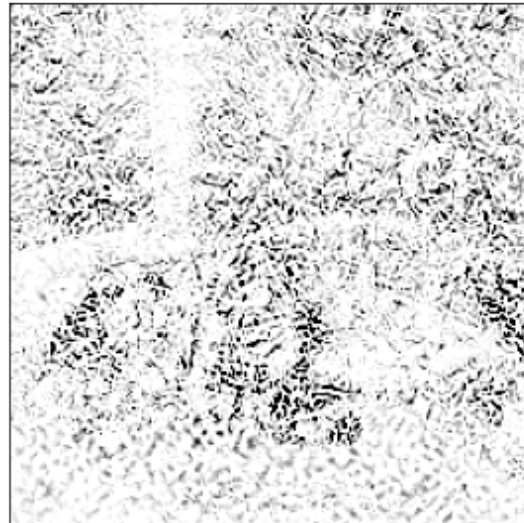


```
[25]: from fastai.callback.captum import *  
  
captum = CaptumInterpretation(learn)  
captum.visualize(cat_example)  
captum.visualize(dog_example)
```

Original Image - (0)



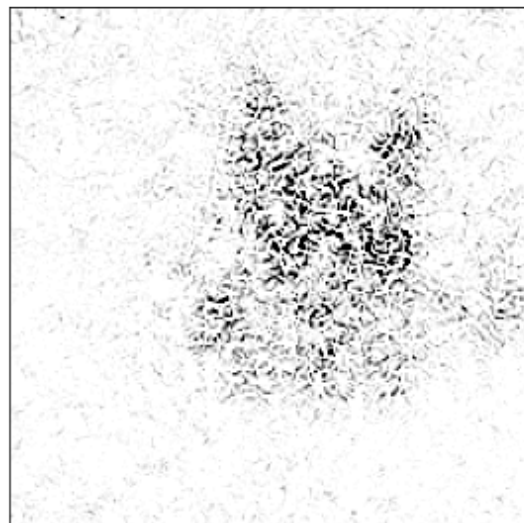
IG



Original Image - (1)

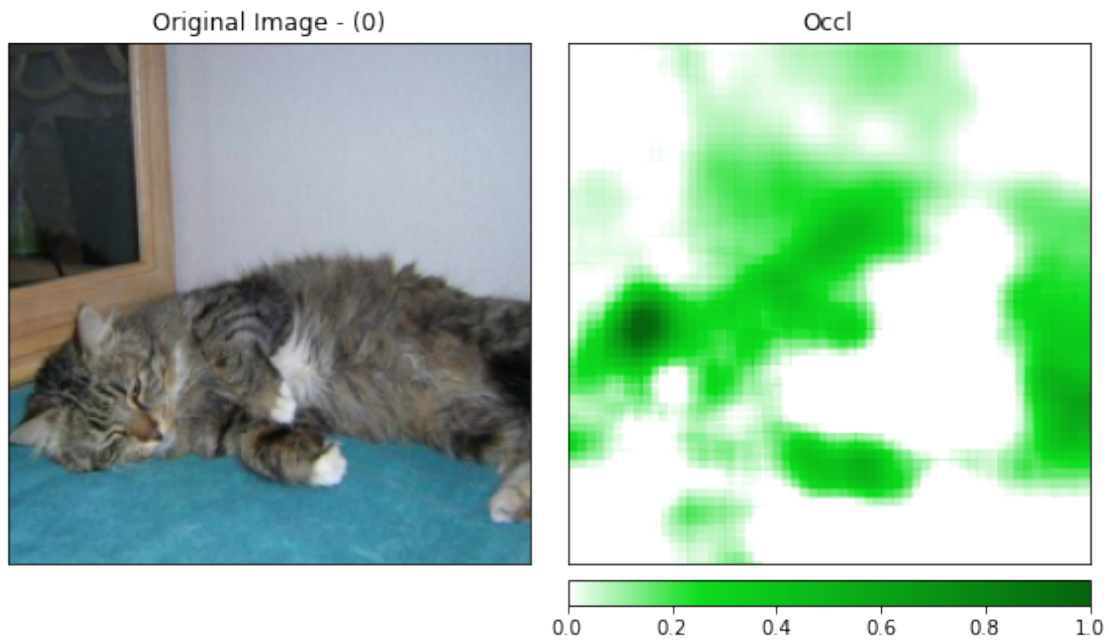


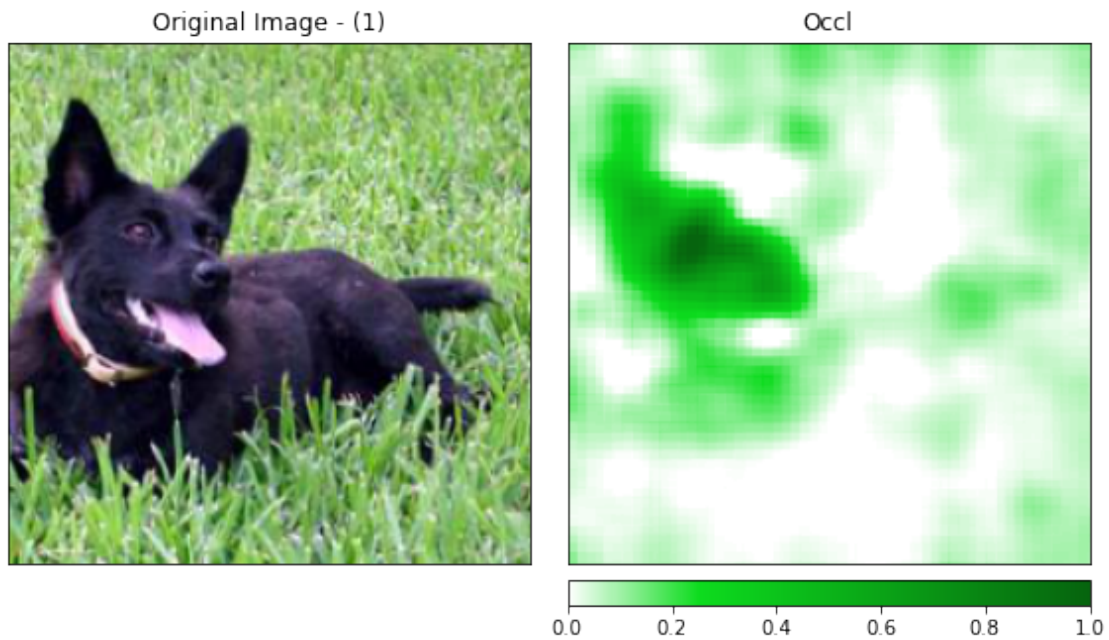
IG



```
[26]: from matplotlib.colors import LinearSegmentedColormap

colors = [(0, "#ffffff"), (0.25, "#0ddc1e"), (1, "#06640e")]
captum = CaptumInterpretation(learn, colors=colors)
captum.visualize(cat_example, metric="Occl")
captum.visualize(dog_example, metric="Occl")
```



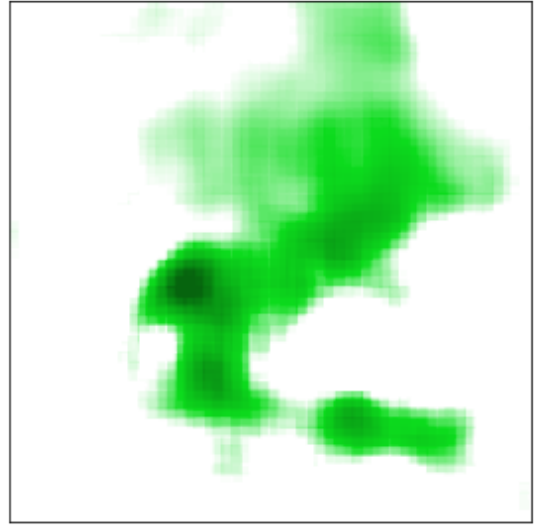


```
[27]: captum = CaptumInterpretation(learn, colors=colors, outlier_perc=2)
captum.visualize(
    cat_example,
    metric="Occl",
)
captum.visualize(
    dog_example,
    metric="Occl",
)
```

Original Image - (0)



Occl



Original Image - (1)



Occl

