


```
1 print('hello world!')
```


“Transfer learning or inductive transfer is a research problem in machine learning that focuses on **storing knowledge gained while solving one problem and applying it to a different but related problem.**”

- Wikipedia

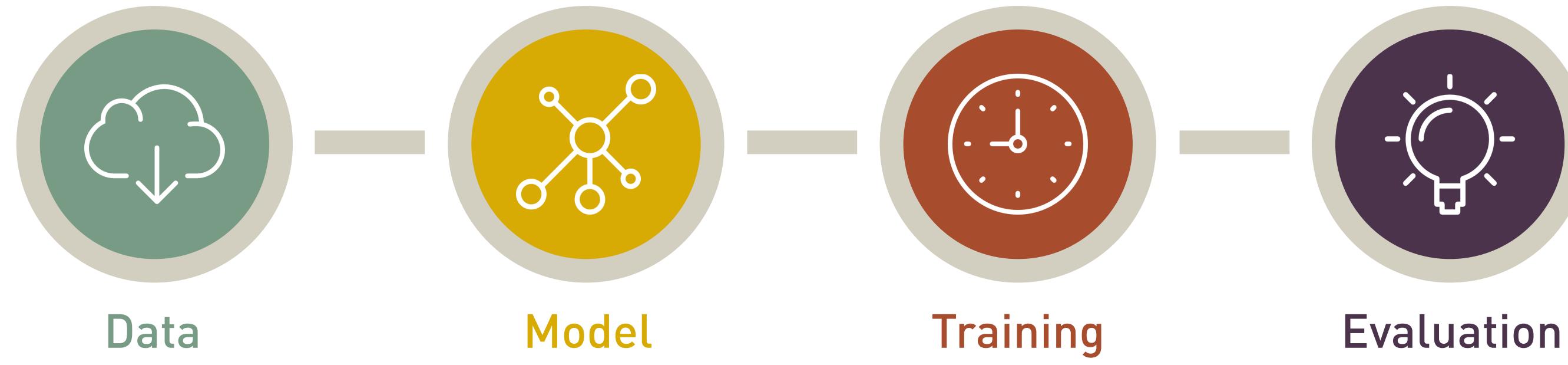
STEP BY STEP

TRANSFER LEARNING

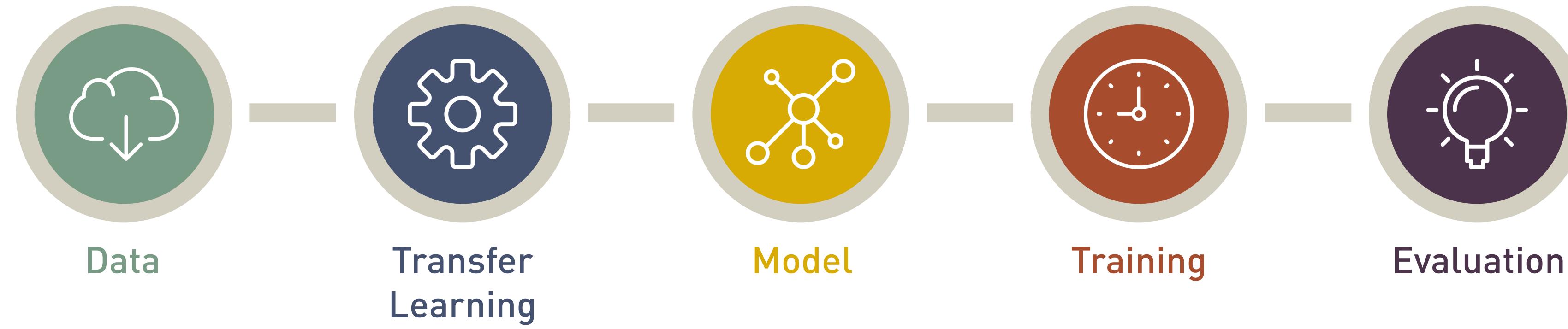
WITH PYTHON AND KERAS

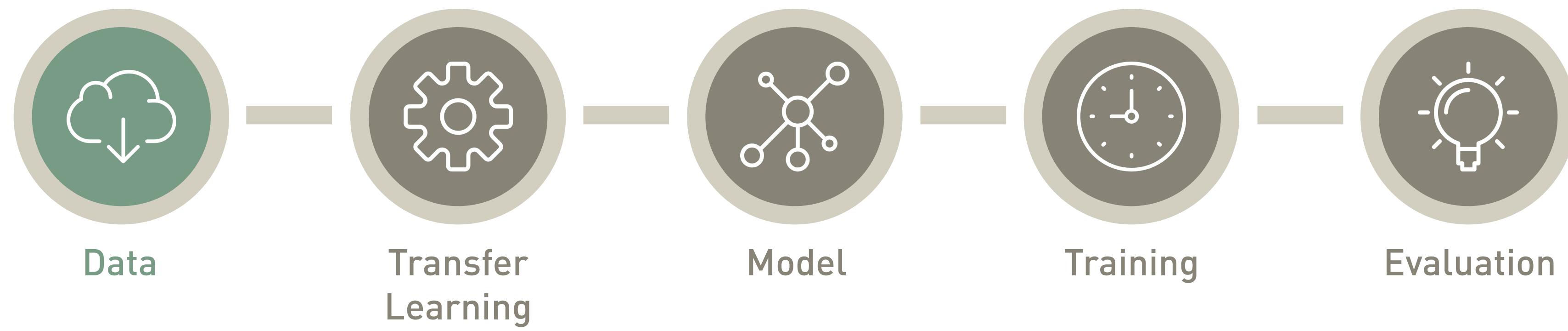


LEARNING PIPELINE

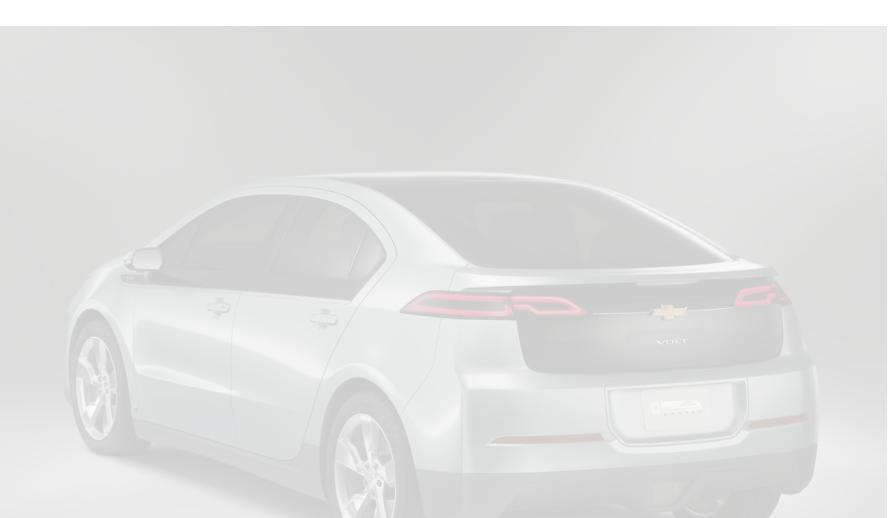
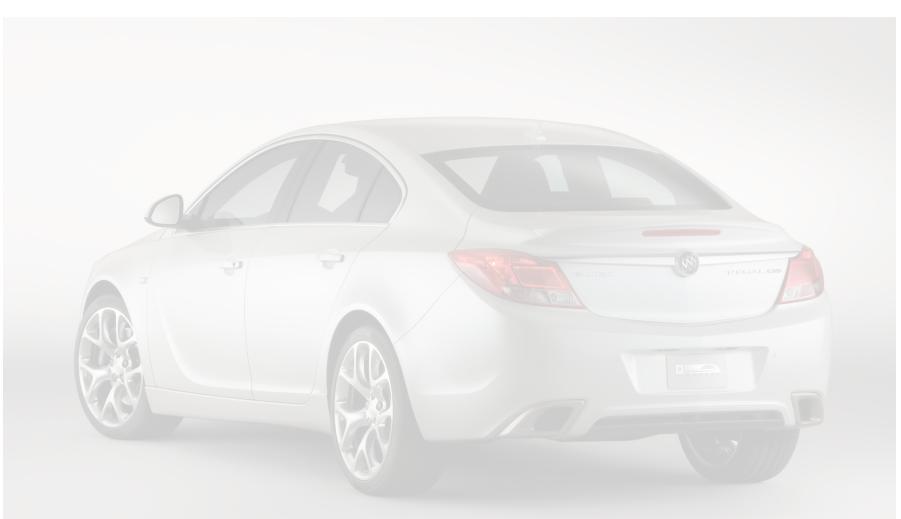


TRANSFER LEARNING PIPELINE

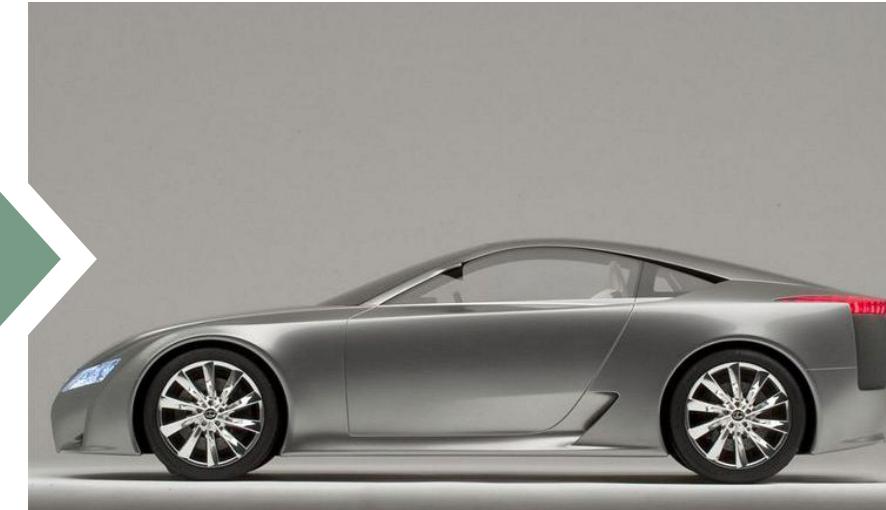








LEFT



RIGHT



FRONT



REAR



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45



```
1 import os
2 import numpy as np
3 from keras.preprocessing.image import load_img, img_to_array
4 from keras.utils.np_utils import to_categorical
5 from sklearn.model_selection import train_test_split
6
7 class ImageDataset:
8     def __init__(self):
9         self.images = []
10        self.labels = []
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
```



```
1 import os
2 import numpy as np
3 from keras.preprocessing.image import load_img, img_to_array
4 from keras.utils.np_utils import to_categorical
5 from sklearn.model_selection import train_test_split
6
7 class ImageDataset:
8     def __init__(self):
9         self.images = []
10        self.labels = []
11
12    def add(self, dir, label):
13        """ Adds the paths to all .jpg files in the given directory
14        to the dataset and labels them. Note: This does not load any
15        files. """
16        files = [os.path.join(dir, f)
17                 for f in os.listdir(dir) if f.endswith('.jpg')]
18        self.images += files
19        self.labels += [label] * len(files)
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
```



```
1 import os
2 import numpy as np
3 from keras.preprocessing.image import load_img, img_to_array
4 from keras.utils.np_utils import to_categorical
5 from sklearn.model_selection import train_test_split
6
7 class ImageDataset:
8     def __init__(self):
9         self.images = []
10        self.labels = []
11
12    def add(self, dir, label):
13        """ Adds the paths to all .jpg files in the given directory
14        to the dataset and labels them. Note: This does not load any
15        files. """
16        files = [os.path.join(dir, f)
17                 for f in os.listdir(dir) if f.endswith('.jpg')]
18        self.images += files
19        self.labels += [label] * len(files)
20
21    def generate(self, test_split=0.33):
22        """ Loads the image files for the stored paths and returns a
23        training/test split of inputs and labels in a format suitable
24        for Keras. """
25        x = np.vstack([self._load_img(img) for img in self.images])
26        y = to_categorical(self.labels)
27        return train_test_split(x, y, test_size=test_split, random_state=42)
28
29    def _load_img(self, path):
30        """ Loads an image from path and converts it to a 4-dimensional
31        tensor to be passed to a Keras model. """
32        img = img_to_array(load_img(path, target_size=(224, 224)))
33        return np.expand_dims(img, axis=0)
34
35
36
37
38
39
40
41
42
43
44
45
```



dataset.py

```
1 import os
2 import numpy as np
3 from keras.preprocessing.image import load_img, img_to_array
4 from keras.utils.np_utils import to_categorical
5 from sklearn.model_selection import train_test_split
6
7 class ImageDataset:
8     def __init__(self):
9         self.images = []
10        self.labels = []
11
12    def add(self, dir, label):
13        """ Adds the paths to all .jpg files in the given directory
14        to the dataset and labels them. Note: This does not load any
15        files. """
16        files = [os.path.join(dir, f)
17                 for f in os.listdir(dir) if f.endswith('.jpg')]
18        self.images += files
19        self.labels += [label] * len(files)
20
21    def generate(self, test_split=0.33):
22        """ Loads the image files for the stored paths and returns a
23        training/test split of inputs and labels in a format suitable
24        for Keras. """
25        x = np.vstack([self._load_img(img) for img in self.images])
26        y = to_categorical(self.labels)
27        return train_test_split(x, y, test_size=test_split, random_state=42)
28
29    def _load_img(self, path):
30        """ Loads an image from path and converts it to a 4-dimensional
31        tensor to be passed to a Keras model. """
32        img = img_to_array(load_img(path, target_size=(224, 224)))
33        return np.expand_dims(img, axis=0)
34
35
36
37
38
39
40
41
42
43
44
45
```

main.py

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
```



dataset.py

```

1 import os
2 import numpy as np
3 from keras.preprocessing.image import load_img, img_to_array
4 from keras.utils.np_utils import to_categorical
5 from sklearn.model_selection import train_test_split
6
7 class ImageDataset:
8     def __init__(self):
9         self.images = []
10        self.labels = []
11
12    def add(self, dir, label):
13        """ Adds the paths to all .jpg files in the given directory
14        to the dataset and labels them. Note: This does not load any
15        files. """
16        files = [os.path.join(dir, f)
17                 for f in os.listdir(dir) if f.endswith('.jpg')]
18        self.images += files
19        self.labels += [label] * len(files)
20
21    def generate(self, test_split=0.33):
22        """ Loads the image files for the stored paths and returns a
23        training/test split of inputs and labels in a format suitable
24        for Keras. """
25        x = np.vstack([self._load_img(img) for img in self.images])
26        y = to_categorical(self.labels)
27        return train_test_split(x, y, test_size=test_split, random_state=42)
28
29    def _load_img(self, path):
30        """ Loads an image from path and converts it to a 4-dimensional
31        tensor to be passed to a Keras model. """
32        img = img_to_array(load_img(path, target_size=(224, 224)))
33        return np.expand_dims(img, axis=0)
34
35
36
37
38
39
40
41
42
43
44
45

```

main.py

```

1 # Define labels and the number of classes.
2 labels = ['Left', 'Right', 'Front', 'Rear']
3 num_classes = len(labels)
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45

```



dataset.py

```

1 import os
2 import numpy as np
3 from keras.preprocessing.image import load_img, img_to_array
4 from keras.utils.np_utils import to_categorical
5 from sklearn.model_selection import train_test_split
6
7 class ImageDataset:
8     def __init__(self):
9         self.images = []
10        self.labels = []
11
12    def add(self, dir, label):
13        """ Adds the paths to all .jpg files in the given directory
14        to the dataset and labels them. Note: This does not load any
15        files. """
16        files = [os.path.join(dir, f)
17                 for f in os.listdir(dir) if f.endswith('.jpg')]
18        self.images += files
19        self.labels += [label] * len(files)
20
21    def generate(self, test_split=0.33):
22        """ Loads the image files for the stored paths and returns a
23        training/test split of inputs and labels in a format suitable
24        for Keras. """
25        x = np.vstack([self._load_img(img) for img in self.images])
26        y = to_categorical(self.labels)
27        return train_test_split(x, y, test_size=test_split, random_state=42)
28
29    def _load_img(self, path):
30        """ Loads an image from path and converts it to a 4-dimensional
31        tensor to be passed to a Keras model. """
32        img = img_to_array(load_img(path, target_size=(224, 224)))
33        return np.expand_dims(img, axis=0)
34
35
36
37
38
39
40
41
42
43
44
45

```

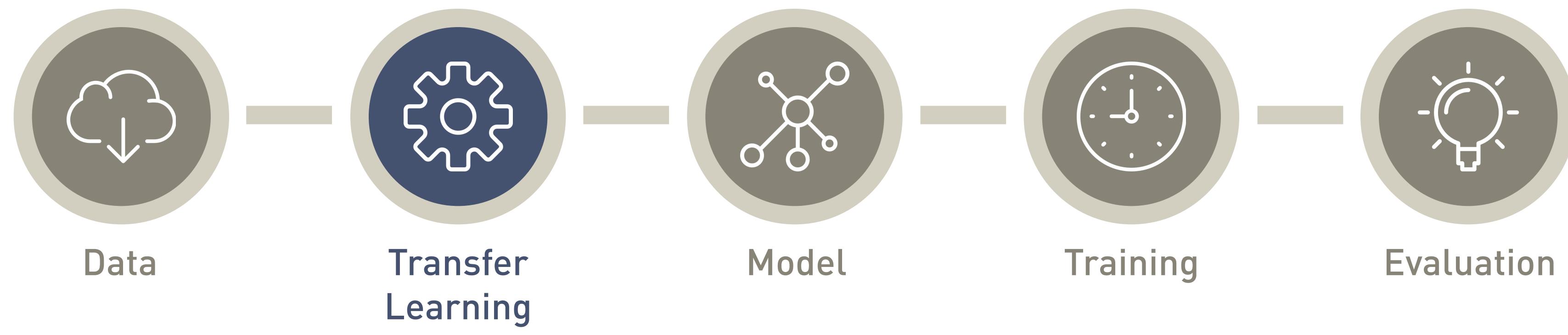
main.py

```

1 # Define labels and the number of classes.
2 labels = ['Left', 'Right', 'Front', 'Rear']
3 num_classes = len(labels)
4
5 import os
6 from dataset import ImageDataset
7
8 # Loop through image folders to load and label images.
9 dataset = ImageDataset()
10 for idx, label in enumerate(labels):
11     path = os.path.join('images', label)
12     dataset.add(path, idx)
13 # Create a train/test split.
14 x_train, x_test, y_train, y_test = dataset.generate(0.25)
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45

```





dataset.py

```

1 import os
2 import numpy as np
3 from keras.preprocessing.image import load_img, img_to_array
4 from keras.utils.np_utils import to_categorical
5 from sklearn.model_selection import train_test_split
6
7 class ImageDataset:
8     def __init__(self):
9         self.images = []
10        self.labels = []
11
12    def add(self, dir, label):
13        """ Adds the paths to all .jpg files in the given directory
14        to the dataset and labels them. Note: This does not load any
15        files. """
16        files = [os.path.join(dir, f)
17                 for f in os.listdir(dir) if f.endswith('.jpg')]
18        self.images += files
19        self.labels += [label] * len(files)
20
21    def generate(self, test_split=0.33):
22        """ Loads the image files for the stored paths and returns a
23        training/test split of inputs and labels in a format suitable
24        for Keras. """
25        x = np.vstack([self._load_img(img) for img in self.images])
26        y = to_categorical(self.labels)
27        return train_test_split(x, y, test_size=test_split, random_state=42)
28
29    def _load_img(self, path):
30        """ Loads an image from path and converts it to a 4-dimensional
31        tensor to be passed to a Keras model. """
32        img = img_to_array(load_img(path, target_size=(224, 224)))
33        return np.expand_dims(img, axis=0)
34
35
36
37
38
39
40
41
42
43
44
45

```

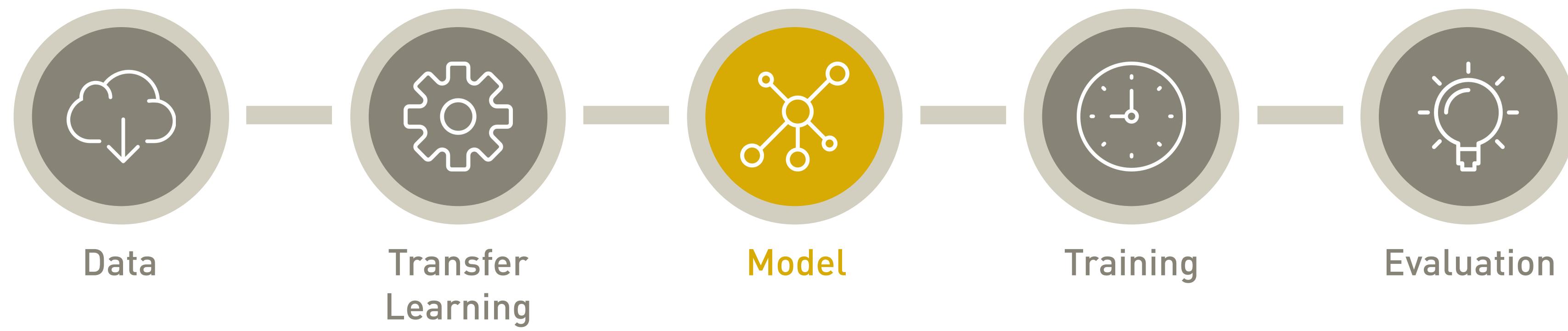
main.py

```

1 # Define labels and the number of classes.
2 labels = ['Left', 'Right', 'Front', 'Rear']
3 num_classes = len(labels)
4
5 import os
6 from dataset import ImageDataset
7
8 # Loop through image folders to load and label images.
9 dataset = ImageDataset()
10 for idx, label in enumerate(labels):
11     path = os.path.join('images', label)
12     dataset.add(path, idx)
13 # Create a train/test split.
14 x_train, x_test, y_train, y_test = dataset.generate(0.25)
15
16 from keras.applications.resnet50 import ResNet50
17 from keras.applications.resnet50 import preprocess_input
18
19 # Run input data through the ResNet50.
20 resnet = ResNet50(weights='imagenet', include_top=False)
21 x_train = resnet.predict(preprocess_input(x_train))
22 x_test = resnet.predict(preprocess_input(x_test))
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45

```





dataset.py

```

1 import os
2 import numpy as np
3 from keras.preprocessing.image import load_img, img_to_array
4 from keras.utils.np_utils import to_categorical
5 from sklearn.model_selection import train_test_split
6
7 class ImageDataset:
8     def __init__(self):
9         self.images = []
10        self.labels = []
11
12    def add(self, dir, label):
13        """ Adds the paths to all .jpg files in the given directory
14        to the dataset and labels them. Note: This does not load any
15        files. """
16        files = [os.path.join(dir, f)
17                 for f in os.listdir(dir) if f.endswith('.jpg')]
18        self.images += files
19        self.labels += [label] * len(files)
20
21    def generate(self, test_split=0.33):
22        """ Loads the image files for the stored paths and returns a
23        training/test split of inputs and labels in a format suitable
24        for Keras. """
25        x = np.vstack([self._load_img(img) for img in self.images])
26        y = to_categorical(self.labels)
27        return train_test_split(x, y, test_size=test_split, random_state=42)
28
29    def _load_img(self, path):
30        """ Loads an image from path and converts it to a 4-dimensional
31        tensor to be passed to a Keras model. """
32        img = img_to_array(load_img(path, target_size=(224, 224)))
33        return np.expand_dims(img, axis=0)
34
35
36
37
38
39
40
41
42
43
44
45

```

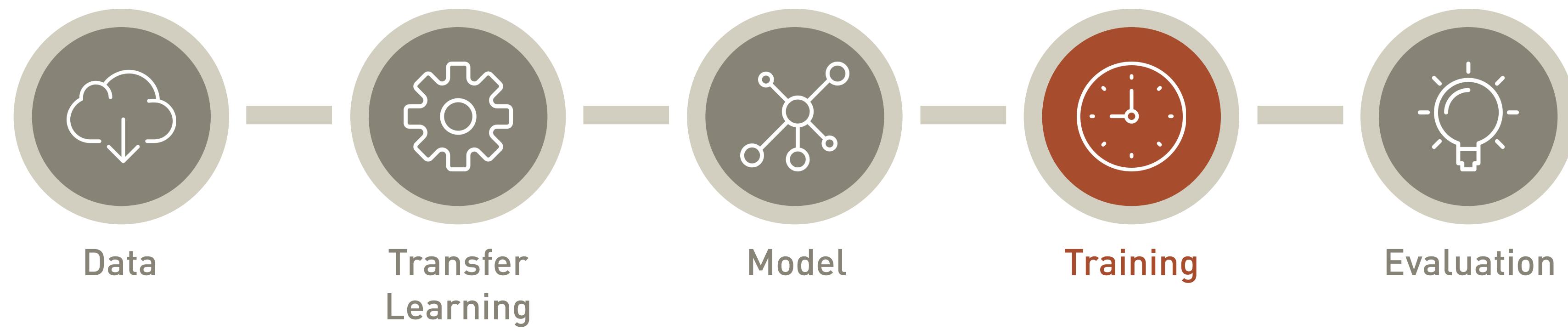
main.py

```

1 # Define labels and the number of classes.
2 labels = ['Left', 'Right', 'Front', 'Rear']
3 num_classes = len(labels)
4
5 import os
6 from dataset import ImageDataset
7
8 # Loop through image folders to load and label images.
9 dataset = ImageDataset()
10 for idx, label in enumerate(labels):
11     path = os.path.join('images', label)
12     dataset.add(path, idx)
13 # Create a train/test split.
14 x_train, x_test, y_train, y_test = dataset.generate(0.25)
15
16 from keras.applications.resnet50 import ResNet50
17 from keras.applications.resnet50 import preprocess_input
18
19 # Run input data through the ResNet50.
20 resnet = ResNet50(weights='imagenet', include_top=False)
21 x_train = resnet.predict(preprocess_input(x_train))
22 x_test = resnet.predict(preprocess_input(x_test))
23
24 from keras.models import Sequential
25 from keras.layers import Dense, Flatten
26
27 # Create a very simple model architecture to be trained on the
28 # output of ResNet50.
29 model = Sequential()
30 model.add(Flatten(input_shape=x_train.shape[1:]))
31 model.add(Dense(num_classes, activation='softmax'))
32 # Compile the model.
33 model.compile(loss='categorical_crossentropy',
34                 optimizer='adam',
35                 metrics=['accuracy'])
36
37
38
39
40
41
42
43
44
45

```





dataset.py

```

1 import os
2 import numpy as np
3 from keras.preprocessing.image import load_img, img_to_array
4 from keras.utils.np_utils import to_categorical
5 from sklearn.model_selection import train_test_split
6
7 class ImageDataset:
8     def __init__(self):
9         self.images = []
10        self.labels = []
11
12    def add(self, dir, label):
13        """ Adds the paths to all .jpg files in the given directory
14        to the dataset and labels them. Note: This does not load any
15        files. """
16        files = [os.path.join(dir, f)
17                 for f in os.listdir(dir) if f.endswith('.jpg')]
18        self.images += files
19        self.labels += [label] * len(files)
20
21    def generate(self, test_split=0.33):
22        """ Loads the image files for the stored paths and returns a
23        training/test split of inputs and labels in a format suitable
24        for Keras. """
25        x = np.vstack([self._load_img(img) for img in self.images])
26        y = to_categorical(self.labels)
27        return train_test_split(x, y, test_size=test_split, random_state=42)
28
29    def _load_img(self, path):
30        """ Loads an image from path and converts it to a 4-dimensional
31        tensor to be passed to a Keras model. """
32        img = img_to_array(load_img(path, target_size=(224, 224)))
33        return np.expand_dims(img, axis=0)
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71

```

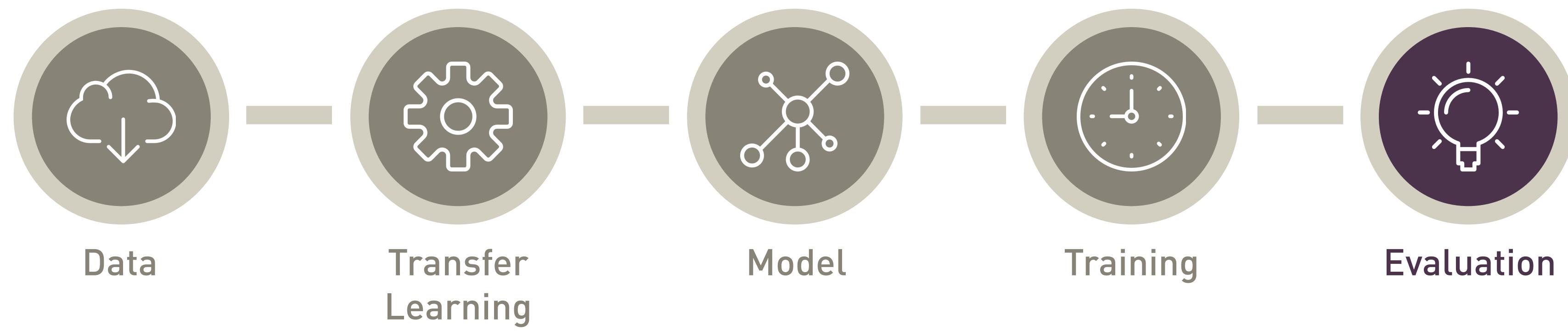
main.py

```

27 # Create a very simple model architecture to be trained on the
28 # output of ResNet50.
29 model = Sequential()
30 model.add(Flatten(input_shape=x_train.shape[1:]))
31 model.add(Dense(num_classes, activation='softmax'))
32 # Compile the model.
33 model.compile(loss='categorical_crossentropy',
34                 optimizer='adam',
35                 metrics=['accuracy'])
36
37 from keras.callbacks import ModelCheckpoint
38
39 # Train the model and store the weights that scored best on the
40 # validation set.
41 checkpoint = ModelCheckpoint(filepath='best.weights.hdf5',
42                             save_best_only=True)
43 progress = model.fit(x_train, y_train,
44                       epochs=500,
45                       batch_size=75,
46                       validation_split=0.33,
47                       callbacks=[checkpoint])
48 # Restore the best weights.
49 model.load_weights('best.weights.hdf5')
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71

```





dataset.py

```

1 import os
2 import numpy as np
3 from keras.preprocessing.image import load_img, img_to_array
4 from keras.utils.np_utils import to_categorical
5 from sklearn.model_selection import train_test_split
6
7 class ImageDataset:
8     def __init__(self):
9         self.images = []
10        self.labels = []
11
12    def add(self, dir, label):
13        """ Adds the paths to all .jpg files in the given directory
14        to the dataset and labels them. Note: This does not load any
15        files. """
16        files = [os.path.join(dir, f)
17                 for f in os.listdir(dir) if f.endswith('.jpg')]
18        self.images += files
19        self.labels += [label] * len(files)
20
21    def generate(self, test_split=0.33):
22        """ Loads the image files for the stored paths and returns a
23        training/test split of inputs and labels in a format suitable
24        for Keras. """
25        x = np.vstack([self._load_img(img) for img in self.images])
26        y = to_categorical(self.labels)
27        return train_test_split(x, y, test_size=test_split, random_state=42)
28
29    def _load_img(self, path):
30        """ Loads an image from path and converts it to a 4-dimensional
31        tensor to be passed to a Keras model. """
32        img = img_to_array(load_img(path, target_size=(224, 224)))
33        return np.expand_dims(img, axis=0)
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71

```

main.py

```

27 # Create a very simple model architecture to be trained on the
28 # output of ResNet50.
29 model = Sequential()
30 model.add(Flatten(input_shape=x_train.shape[1:]))
31 model.add(Dense(num_classes, activation='softmax'))
32 # Compile the model.
33 model.compile(loss='categorical_crossentropy',
34                 optimizer='adam',
35                 metrics=['accuracy'])
36
37 from keras.callbacks import ModelCheckpoint
38
39 # Train the model and store the weights that scored best on the
40 # validation set.
41 checkpoint = ModelCheckpoint(filepath='best.weights.hdf5',
42                               save_best_only=True)
43 progress = model.fit(x_train, y_train,
44                       epochs=500,
45                       batch_size=75,
46                       validation_split=0.33,
47                       callbacks=[checkpoint])
48 # Restore the best weights.
49 model.load_weights('best.weights.hdf5')
50
51 import numpy as np
52 import matplotlib.pyplot as plt
53
54 # Print best validation accuracy.
55 print(np.max(progress.history['val_acc']))
56 # Plot the training and validation loss.
57 plt.plot(progress.history['loss'])
58 plt.plot(progress.history['val_loss'])
59 plt.xlabel('Epoch')
60 plt.ylabel('Loss')
61 plt.title('Training and Validation Loss')
62 leg = plt.legend(['Training Loss', 'Validation Loss'])
63 plt.show()
64
65
66
67
68
69
70
71

```



85.02%
VALIDATION ACCURACY



TRAINING LOSS

VALIDATION LOSS

3

2.5

2

1.5

1

0.5



dataset.py

```

1 import os
2 import numpy as np
3 from keras.preprocessing.image import load_img, img_to_array
4 from keras.utils.np_utils import to_categorical
5 from sklearn.model_selection import train_test_split
6
7 class ImageDataset:
8     def __init__(self):
9         self.images = []
10        self.labels = []
11
12    def add(self, dir, label):
13        """ Adds the paths to all .jpg files in the given directory
14        to the dataset and labels them. Note: This does not load any
15        files. """
16        files = [os.path.join(dir, f)
17                 for f in os.listdir(dir) if f.endswith('.jpg')]
18        self.images += files
19        self.labels += [label] * len(files)
20
21    def generate(self, test_split=0.33):
22        """ Loads the image files for the stored paths and returns a
23        training/test split of inputs and labels in a format suitable
24        for Keras. """
25        x = np.vstack([self._load_img(img) for img in self.images])
26        y = to_categorical(self.labels)
27        return train_test_split(x, y, test_size=test_split, random_state=42)
28
29    def _load_img(self, path):
30        """ Loads an image from path and converts it to a 4-dimensional
31        tensor to be passed to a Keras model. """
32        img = img_to_array(load_img(path, target_size=(224, 224)))
33        return np.expand_dims(img, axis=0)
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71

```

main.py

```

27 # Create a very simple model architecture to be trained on the
28 # output of ResNet50.
29 model = Sequential()
30 model.add(Flatten(input_shape=x_train.shape[1:]))
31 model.add(Dropout(0.75))
32 model.add(Dense(num_classes, activation='softmax'))
33 # Compile the model.
34 model.compile(loss='categorical_crossentropy',
35                 optimizer='adam',
36                 metrics=['accuracy'])
37
38 from keras.callbacks import ModelCheckpoint
39
40 # Train the model and store the weights that scored best on the
41 # validation set.
42 checkpoint = ModelCheckpoint(filepath='best.weights.hdf5',
43                             save_best_only=True)
44 progress = model.fit(x_train, y_train,
45                       epochs=500,
46                       batch_size=75,
47                       validation_split=0.33,
48                       callbacks=[checkpoint])
49 # Restore the best weights.
50 model.load_weights('best.weights.hdf5')
51
52 import numpy as np
53 import matplotlib.pyplot as plt
54
55 # Print best validation accuracy.
56 print(np.max(progress.history['val_acc']))
57 # Plot the training and validation loss.
58 plt.plot(progress.history['loss'])
59 plt.plot(progress.history['val_loss'])
60 plt.xlabel('Epoch')
61 plt.ylabel('Loss')
62 plt.title('Training and Validation Loss')
63 leg = plt.legend(['Training Loss', 'Validation Loss'])
64
65
66
67
68
69
70
71

```

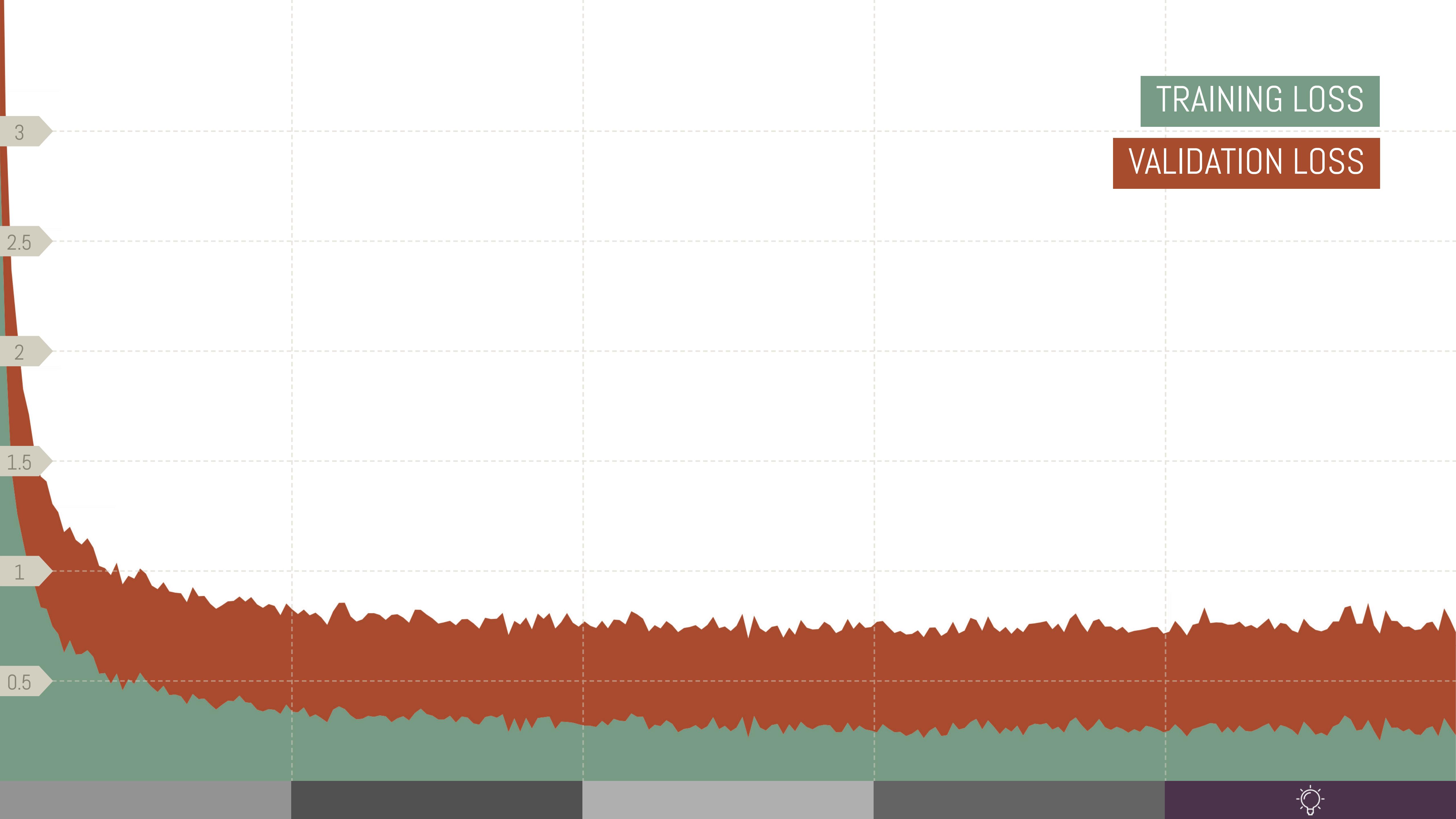


85.99%
VALIDATION ACCURACY



TRAINING LOSS

VALIDATION LOSS



dataset.py

```

1 import os
2 import numpy as np
3 from keras.preprocessing.image import load_img, img_to_array
4 from keras.utils.np_utils import to_categorical
5 from sklearn.model_selection import train_test_split
6
7 class ImageDataset:
8     def __init__(self):
9         self.images = []
10        self.labels = []
11
12    def add(self, dir, label):
13        """ Adds the paths to all .jpg files in the given directory
14        to the dataset and labels them. Note: This does not load any
15        files. """
16        files = [os.path.join(dir, f)
17                 for f in os.listdir(dir) if f.endswith('.jpg')]
18        self.images += files
19        self.labels += [label] * len(files)
20
21    def generate(self, test_split=0.33):
22        """ Loads the image files for the stored paths and returns a
23        training/test split of inputs and labels in a format suitable
24        for Keras. """
25        x = np.vstack([self._load_img(img) for img in self.images])
26        y = to_categorical(self.labels)
27        return train_test_split(x, y, test_size=test_split, random_state=42)
28
29    def _load_img(self, path):
30        """ Loads an image from path and converts it to a 4-dimensional
31        tensor to be passed to a Keras model. """
32        img = img_to_array(load_img(path, target_size=(224, 224)))
33        return np.expand_dims(img, axis=0)
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98

```

main.py

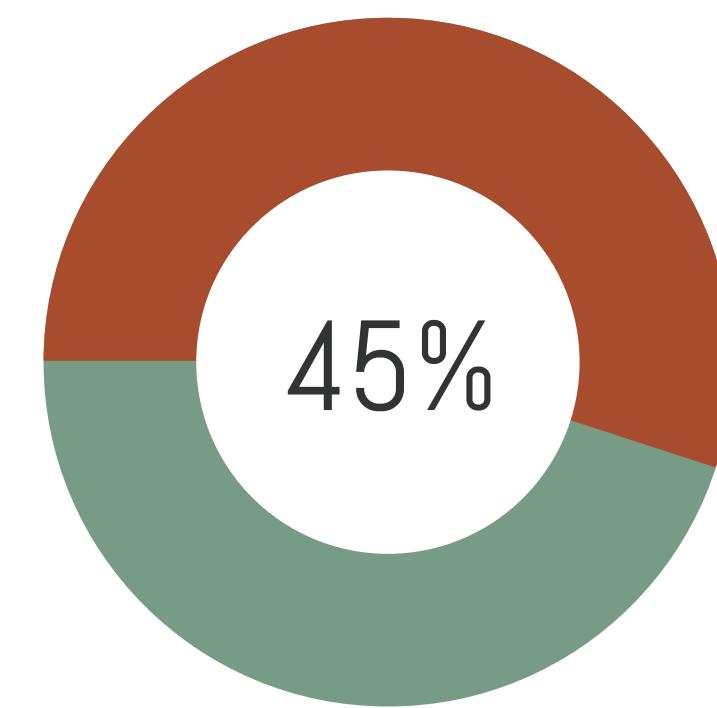
```

54 # Print best validation accuracy.
55 print(np.max(progress.history['val_acc']))
56 # Plot the training and validation loss.
57 plt.plot(progress.history['loss'])
58 plt.plot(progress.history['val_loss'])
59 plt.xlabel('Epoch')
60 plt.ylabel('Loss')
61 plt.title('Training and Validation Loss')
62 leg = plt.legend(['Training Loss', 'Validation Loss'])
63 plt.show()
64
65 from sklearn.metrics import confusion_matrix
66
67 # Print the confusion matrix to see where most of the errors happen.
68 validation_split = round(0.33 * len(x_train))
69 predictions = model.predict_classes(x_train[-validation_split:])
70 expectations = np.argmax(y_train[-validation_split:], axis=1)
71 matrix = confusion_matrix(expectations, predictions)
72 print(matrix)
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98

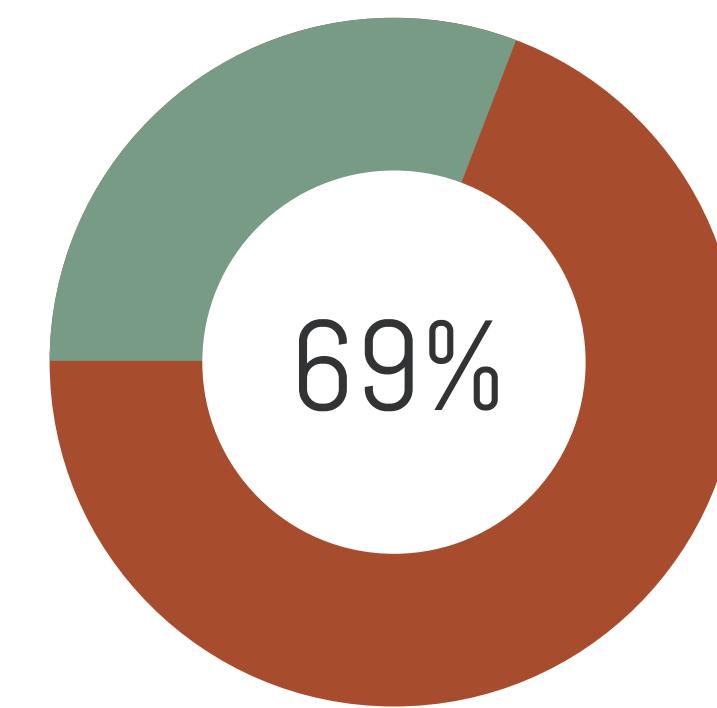
```



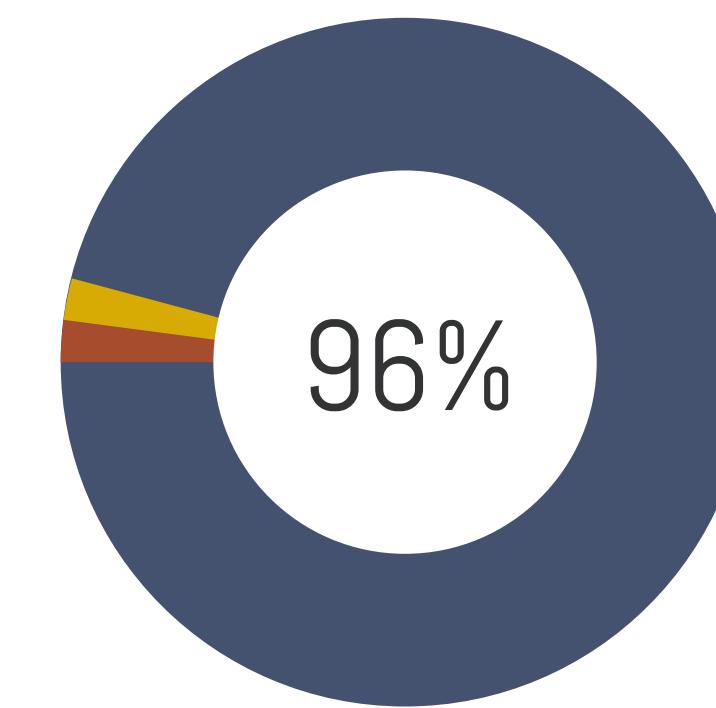
CONFUSION



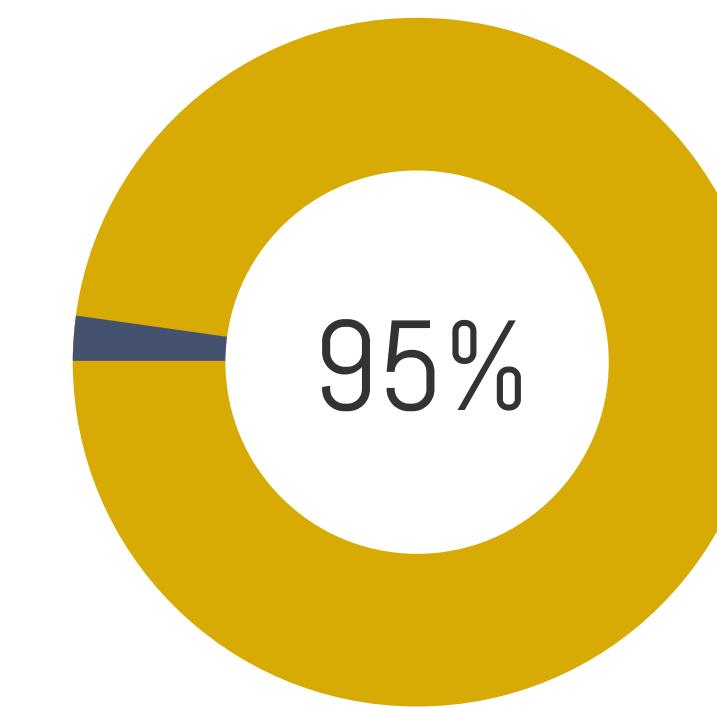
LEFT



RIGHT



FRONT



REAR



dataset.py

```

1 import os
2 import numpy as np
3 from keras.preprocessing.image import load_img, img_to_array
4 from keras.utils.np_utils import to_categorical
5 from sklearn.model_selection import train_test_split
6
7 class ImageDataset:
8     def __init__(self):
9         self.images = []
10        self.labels = []
11
12    def add(self, dir, label):
13        """ Adds the paths to all .jpg files in the given directory
14        to the dataset and labels them. Note: This does not load any
15        files. """
16        files = [os.path.join(dir, f)
17                 for f in os.listdir(dir) if f.endswith('.jpg')]
18        self.images += files
19        self.labels += [label] * len(files)
20
21    def generate(self, test_split=0.33):
22        """ Loads the image files for the stored paths and returns a
23        training/test split of inputs and labels in a format suitable
24        for Keras. """
25        x = np.vstack([self._load_img(img) for img in self.images])
26        y = to_categorical(self.labels)
27        return train_test_split(x, y, test_size=test_split, random_state=42)
28
29    def _load_img(self, path):
30        """ Loads an image from path and converts it to a 4-dimensional
31        tensor to be passed to a Keras model. """
32        img = img_to_array(load_img(path, target_size=(224, 224)))
33        return np.expand_dims(img, axis=0)
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98

```

main.py

```

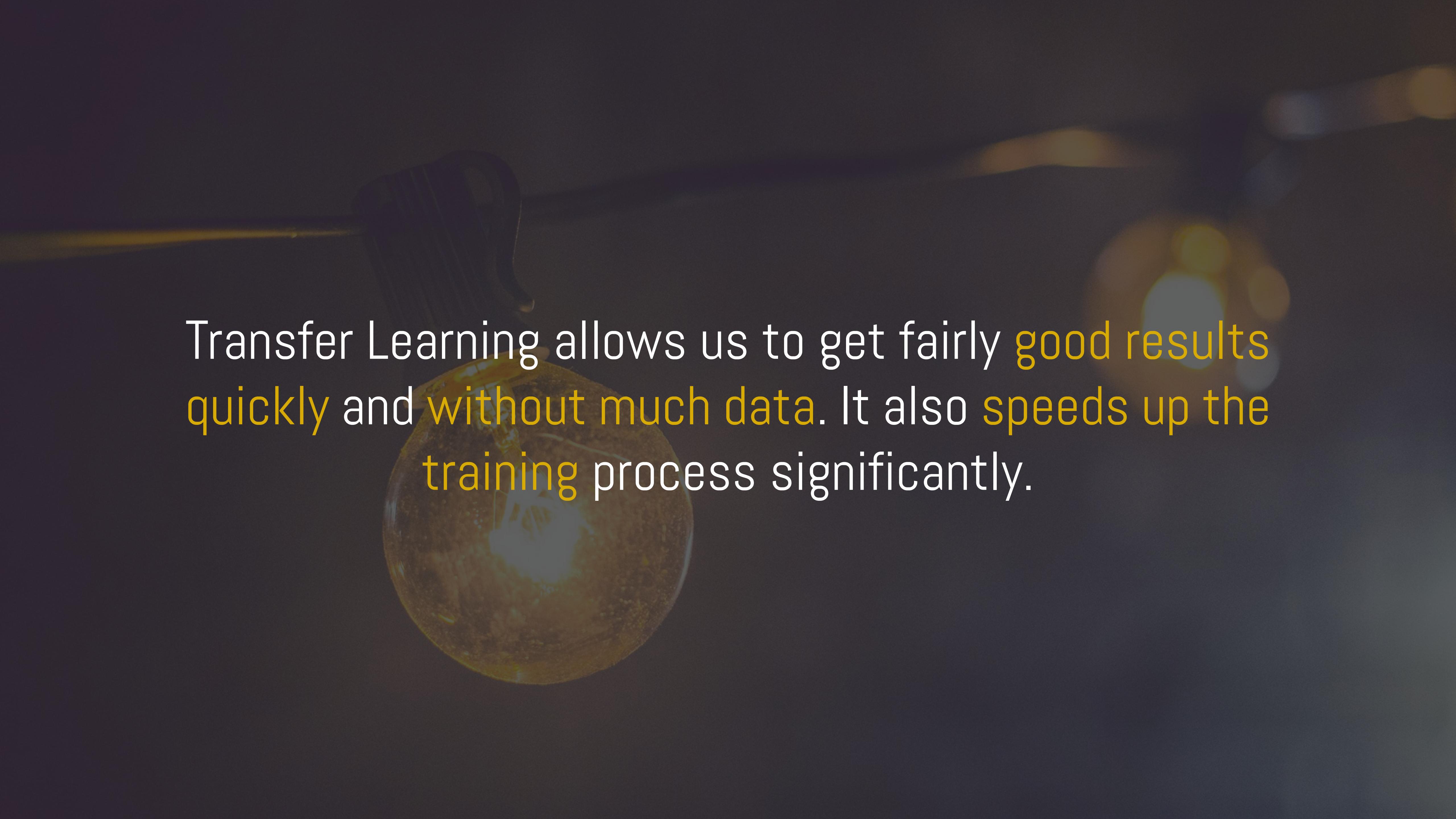
54 # Print best validation accuracy.
55 print(np.max(progress.history['val_acc']))
56 # Plot the training and validation loss.
57 plt.plot(progress.history['loss'])
58 plt.plot(progress.history['val_loss'])
59 plt.xlabel('Epoch')
60 plt.ylabel('Loss')
61 plt.title('Training and Validation Loss')
62 leg = plt.legend(['Training Loss', 'Validation Loss'])
63 plt.show()
64
65 from sklearn.metrics import confusion_matrix
66
67 # Print the confusion matrix to see where most of the errors happen.
68 validation_split = round(0.33 * len(x_train))
69 predictions = model.predict_classes(x_train[-validation_split:])
70 expectations = np.argmax(y_train[-validation_split:], axis=1)
71 matrix = confusion_matrix(expectations, predictions)
72 print(matrix)
73
74 # Evaluate the model on the test set.
75 loss, accuracy = model.evaluate(x_test, y_test)
76 print('Test Accuracy: {}'.format(accuracy))
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98

```



81.29%
TEST ACCURACY





Transfer Learning allows us to get fairly good results quickly and without much data. It also speeds up the training process significantly.

dataset.py

```
1 import os
2 import numpy as np
3 from array
4 from
5 from
6
7 class ImageDataset:
8     def __init__(self):
9         self.images = []
10        self.labels = []
11
12    def add(self, dir, label):
13        """ Adds the paths to all .jpg files in the given directory to the dataset and labels them. Note: This does not load files. """
14        files = [os.path.join(dir, f) for f in os.listdir(dir) if f.endswith('.jpg')]
15        self.images += files
16        self.labels += [label] * len(files)
17
18    def generate(self, test_split=0.33):
19        """ Loads the image files for the stored paths and returns a training/test split of inputs and labels in a format suitable for Keras. """
20        x = np.vstack([self._load_img(img) for img in self.images])
21        y = to_categorical(self.labels)
22        return train_test_split(x, y, test_size=test_split, random_state=42)
23
24    def _load_img(self, path):
25        """ Loads an image from path and converts it to a 4-dimensional tensor to be passed to a Keras model. """
26        img = img_to_array(load_img(path, target_size=(224, 224)))
27        return np.expand_dims(img, axis=0)
```



main.py

```
54    # Print best validation accuracy.
55    print(np.max(progress.history['val_acc']))
56    # Plot the training and validation loss.
57    plt.plot(progress.history['loss'])
58    plt.plot(progress.history['val_loss'])
59    plt.xlabel('Epoch')
60    plt.ylabel('Loss')
61    plt.title('Training and Validation Loss')
62    leg = plt.legend(['Training Loss', 'Validation Loss'])
63    plt.show()
```



```
64    # Print the confusion matrix to see where most of the errors happen.
65    validation_split = round(0.33 * len(x_train))
66    x_validation = x_train[-validation_split:]
67    y_validation = np.argmax(y_train[-validation_split:], axis=1)
68    confusion_matrix(expectations, predictions)
```

```
69    # Train the model on the test set.
70    accuracy = model.evaluate(x_test, y_test)
71    print('Accuracy: {}'.format(accuracy))
```

 JOHANNESSTRICKER/TRANSFER-LEARNING.GIT

 JOHANNESSTRICKER@GMX.NET

 JOHANNES-STRICKER