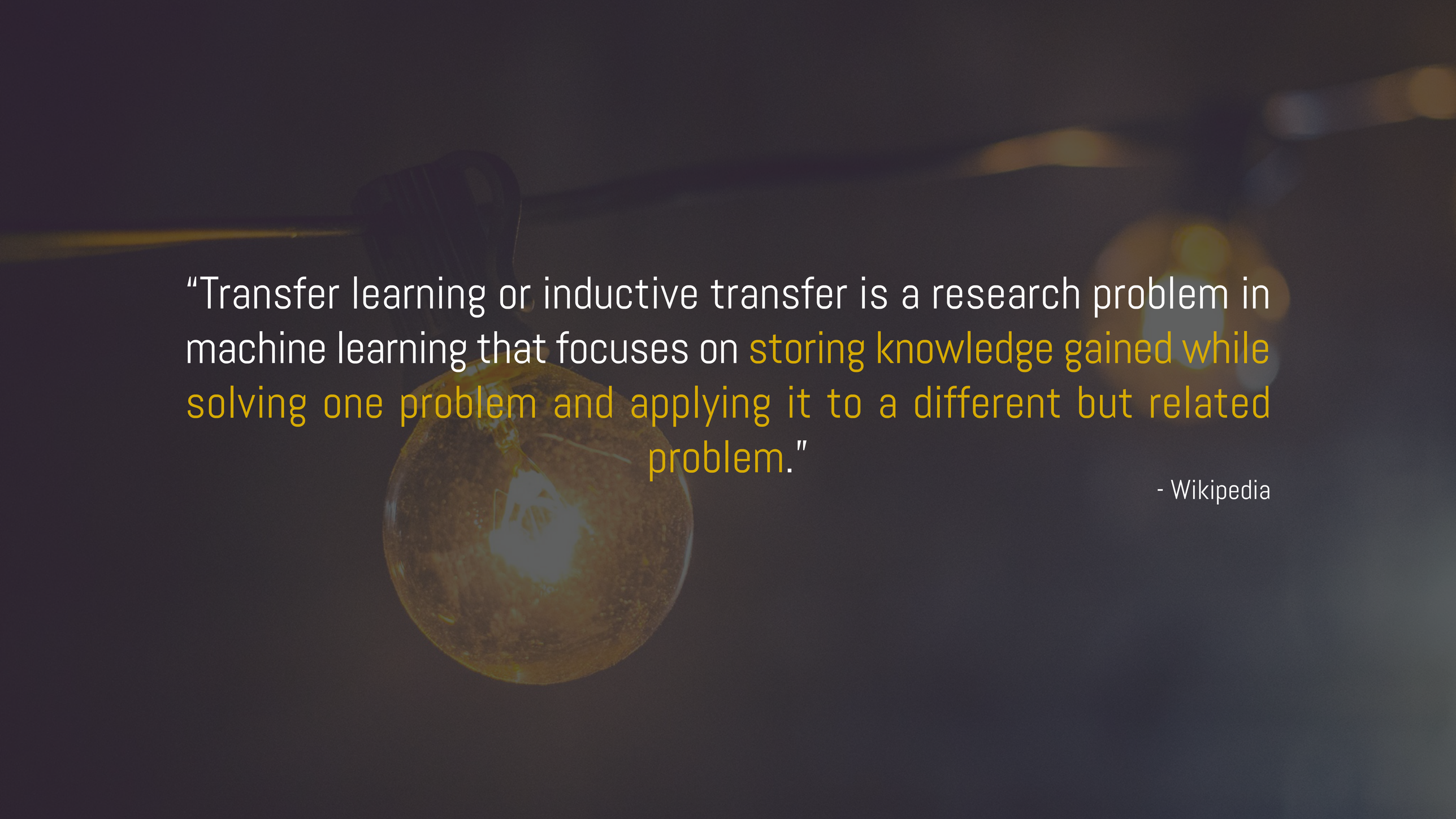
hello_world.py

```
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."

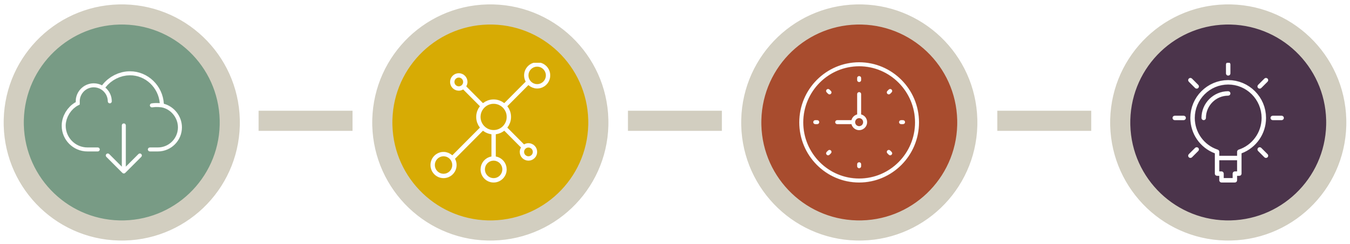
- Wikipedia

STEP BY STEP

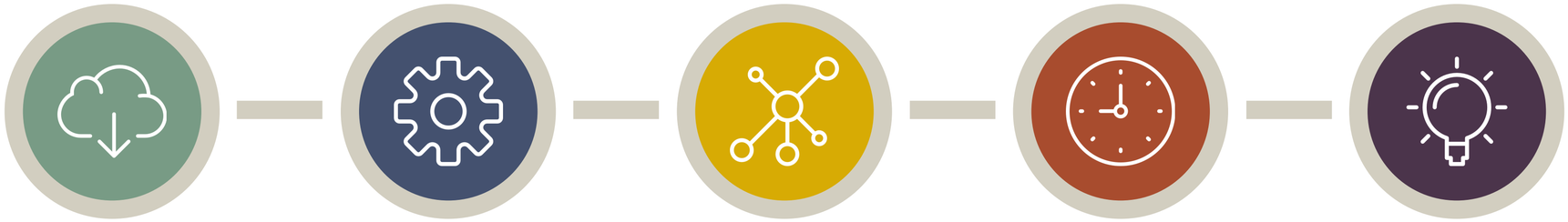
TRANSFER LEARNING

WITH PYTHON AND KERAS

LEARNING PIPELINE



TRANSFER LEARNING PIPELINE




Data
Transfer Learning
Model
Training
Evaluation


Instrument panel
GPS navigation system
Windshield wiper lever
Select lever
Parking brake
Brake pedal
Gas pedal
Turn signal lever
Passenger's seat
Driver's seat



T181OX 178
TOYOTA
V6
HIGHLANDER

Instrument panel
GPS navigation system
Windshield wiper lever
Select lever
Parking Brake
Brake pedal
Gas pedal
Turn signal lever
Passenger's seat
Driver's seat



IN A 6800



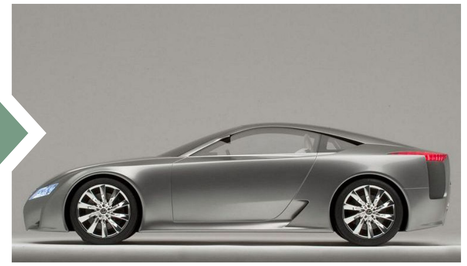
M LC 3534





M LD 1948


TUXEDO

LEFT				
RIGHT				
FRONT				
REAR				

dataset.py

dataset.py

```python
import os
import numpy as np
from keras.preprocessing.image import load_img, img_to_array
from keras.utils.np_utils import to_categorical
from sklearn.model_selection import train_test_split

class ImageDataset:
  def __init__(self):
    self.images = []
    self.labels = []
```

dataset.py

```python
import os
import numpy as np
from keras.preprocessing.image import load_img, img_to_array
from keras.utils.np_utils import to_categorical
from sklearn.model_selection import train_test_split

class ImageDataset:
  def __init__(self):
    self.images = []
    self.labels = []

  def add(self, dir, label):
    “”” Adds the paths to all .jpg files in the given directory
    to the dataset and labels them. Note: This does not load any
    files. “””
    files = [os.path.join(dir, f)
             for f in os.listdir(dir) if f.endswith(‘.jpg’)]
    self.images += files
    self.labels += [label] * len(files)
```

dataset.py

```python
import os
import numpy as np
from keras.preprocessing.image import load_img, img_to_array
from keras.utils.np_utils import to_categorical
from sklearn.model_selection import train_test_split

class ImageDataset:
  def __init__(self):
    self.images = []
    self.labels = []

  def add(self, dir, label):
    “”” Adds the paths to all .jpg files in the given directory
    to the dataset and labels them. Note: This does not load any
    files. “””
    files = [os.path.join(dir, f)
             for f in os.listdir(dir) if f.endswith(‘.jpg’)]
    self.images += files
    self.labels += [label] * len(files)

  def generate(self, test_split=0.33):
    “”” Loads the image files for the stored paths and returns a
    training/test split of inputs and labels in a format suitable
    for Keras. “””
    x = np.vstack([self._load_img(img) for img in self.images])
    y = to_categorical(self.labels)
    return train_test_split(x, y, test_size=test_split, random_state=42)

  def _load_img(self, path):
    “”” Loads an image from path and converts it to a 4-dimensional
    tensor to be passed to a Keras model. “””
    img = img_to_array(load_img(path, target_size=(224, 224)))
    return np.expand_dims(img, axis=0)
```

dataset.py

```python
import os
import numpy as np
from keras.preprocessing.image import load_img, img_to_array
from keras.utils.np_utils import to_categorical
from sklearn.model_selection import train_test_split

class ImageDataset:
  def __init__(self):
    self.images = []
    self.labels = []

  def add(self, dir, label):
    “”” Adds the paths to all .jpg files in the given directory
    to the dataset and labels them. Note: This does not load any
    files. “””
    files = [os.path.join(dir, f)
             for f in os.listdir(dir) if f.endswith(‘.jpg’)]
    self.images += files
    self.labels += [label] * len(files)

  def generate(self, test_split=0.33):
    “”” Loads the image files for the stored paths and returns a
    training/test split of inputs and labels in a format suitable
    for Keras. “””
    x = np.vstack([self._load_img(img) for img in self.images])
    y = to_categorical(self.labels)
    return train_test_split(x, y, test_size=test_split, random_state=42)

  def _load_img(self, path):
    “”” Loads an image from path and converts it to a 4-dimensional
    tensor to be passed to a Keras model. “””
    img = img_to_array(load_img(path, target_size=(224, 224)))
    return np.expand_dims(img, axis=0)
```

main.py

dataset.py

```python
import os
import numpy as np
from keras.preprocessing.image import load_img, img_to_array
from keras.utils.np_utils import to_categorical
from sklearn.model_selection import train_test_split

class ImageDataset:
  def __init__(self):
    self.images = []
    self.labels = []

  def add(self, dir, label):
    “”” Adds the paths to all .jpg files in the given directory
    to the dataset and labels them. Note: This does not load any
    files. “””
    files = [os.path.join(dir, f)
             for f in os.listdir(dir) if f.endswith(‘.jpg’)]
    self.images += files
    self.labels += [label] * len(files)

  def generate(self, test_split=0.33):
    “”” Loads the image files for the stored paths and returns a
    training/test split of inputs and labels in a format suitable
    for Keras. “””
    x = np.vstack([self._load_img(img) for img in self.images])
    y = to_categorical(self.labels)
    return train_test_split(x, y, test_size=test_split, random_state=42)

  def _load_img(self, path):
    “”” Loads an image from path and converts it to a 4-dimensional
    tensor to be passed to a Keras model. “””
    img = img_to_array(load_img(path, target_size=(224, 224)))
    return np.expand_dims(img, axis=0)
```

main.py

```python
# Define labels and the number of classes.
labels = [‘Left’, ‘Right’, ‘Front’, ‘Rear’]
num_classes = len(labels)
```

dataset.py

```python
import os
import numpy as np
from keras.preprocessing.image import load_img, img_to_array
from keras.utils.np_utils import to_categorical
from sklearn.model_selection import train_test_split

class ImageDataset:
  def __init__(self):
    self.images = []
    self.labels = []

  def add(self, dir, label):
    “”” Adds the paths to all .jpg files in the given directory
    to the dataset and labels them. Note: This does not load any
    files. “””
    files = [os.path.join(dir, f)
             for f in os.listdir(dir) if f.endswith(‘.jpg’)]
    self.images += files
    self.labels += [label] * len(files)

  def generate(self, test_split=0.33):
    “”” Loads the image files for the stored paths and returns a
    training/test split of inputs and labels in a format suitable
    for Keras. “””
    x = np.vstack([self._load_img(img) for img in self.images])
    y = to_categorical(self.labels)
    return train_test_split(x, y, test_size=test_split, random_state=42)

  def _load_img(self, path):
    “”” Loads an image from path and converts it to a 4-dimensional
    tensor to be passed to a Keras model. “””
    img = img_to_array(load_img(path, target_size=(224, 224)))
    return np.expand_dims(img, axis=0)
```

main.py

```python
# Define labels and the number of classes.
labels = [‘Left’, ‘Right’, ‘Front’, ‘Rear’]
num_classes = len(labels)

import os
from dataset import ImageDataset

# Loop through image folders to load and label images.
dataset = ImageDataset()
for idx, label in enumerate(labels):
  path = os.path.join(‘images’, label)
  dataset.add(path, idx)
# Create a train/test split.
x_train, x_test, y_train, y_test = dataset.generate(0.25)
```

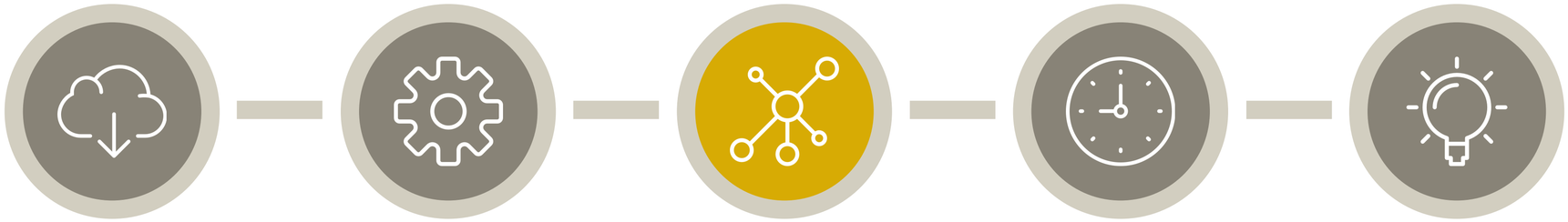

Data
Transfer Learning
Model
Training
Evaluation

dataset.py

```python
import os
import numpy as np
from keras.preprocessing.image import load_img, img_to_array
from keras.utils.np_utils import to_categorical
from sklearn.model_selection import train_test_split

class ImageDataset:
  def __init__(self):
    self.images = []
    self.labels = []

  def add(self, dir, label):
    """ Adds the paths to all .jpg files in the given directory
    to the dataset and labels them. Note: This does not load any
    files. """
    files = [os.path.join(dir, f)
             for f in os.listdir(dir) if f.endswith('.jpg')]
    self.images += files
    self.labels += [label] * len(files)

  def generate(self, test_split=0.33):
    """ Loads the image files for the stored paths and returns a
    training/test split of inputs and labels in a format suitable
    for Keras. """
    x = np.vstack([self._load_img(img) for img in self.images])
    y = to_categorical(self.labels)
    return train_test_split(x, y, test_size=test_split, random_state=42)

  def _load_img(self, path):
    """ Loads an image from path and converts it to a 4-dimensional
    tensor to be passed to a Keras model. """
    img = img_to_array(load_img(path, target_size=(224, 224)))
    return np.expand_dims(img, axis=0)
```

main.py

```python
# Define labels and the number of classes.
labels = ['Left', 'Right', 'Front', 'Rear']
num_classes = len(labels)

import os
from dataset import ImageDataset

# Loop through image folders to load and label images.
dataset = ImageDataset()
for idx, label in enumerate(labels):
  path = os.path.join('images', label)
  dataset.add(path, idx)
# Create a train/test split.
x_train, x_test, y_train, y_test = dataset.generate(0.25)

from keras.applications.resnet50 import ResNet50
from keras.applications.resnet50 import preprocess_input

# Run input data through the ResNet50.
resnet = ResNet50(weights='imagenet', include_top=False)
x_train = resnet.predict(preprocess_input(x_train))
x_test = resnet.predict(preprocess_input(x_test))
```


Data
Transfer Learning
Model
Training
Evaluation

dataset.py

```python
import os
import numpy as np
from keras.preprocessing.image import load_img, img_to_array
from keras.utils.np_utils import to_categorical
from sklearn.model_selection import train_test_split

class ImageDataset:
  def __init__(self):
    self.images = []
    self.labels = []

  def add(self, dir, label):
    “”” Adds the paths to all .jpg files in the given directory
    to the dataset and labels them. Note: This does not load any
    files. “””
    files = [os.path.join(dir, f)
             for f in os.listdir(dir) if f.endswith(‘.jpg’)]
    self.images += files
    self.labels += [label] * len(files)

  def generate(self, test_split=0.33):
    “”” Loads the image files for the stored paths and returns a
    training/test split of inputs and labels in a format suitable
    for Keras. “””
    x = np.vstack([self._load_img(img) for img in self.images])
    y = to_categorical(self.labels)
    return train_test_split(x, y, test_size=test_split, random_state=42)

  def _load_img(self, path):
    “”” Loads an image from path and converts it to a 4-dimensional
    tensor to be passed to a Keras model. “””
    img = img_to_array(load_img(path, target_size=(224, 224)))
    return np.expand_dims(img, axis=0)
```

main.py

```python
# Define labels and the number of classes.
labels = [‘Left’, ‘Right’, ‘Front’, ‘Rear’]
num_classes = len(labels)

import os
from dataset import ImageDataset

# Loop through image folders to load and label images.
dataset = ImageDataset()
for idx, label in enumerate(labels):
  path = os.path.join(‘images’, label)
  dataset.add(path, idx)
# Create a train/test split.
x_train, x_test, y_train, y_test = dataset.generate(0.25)

from keras.applications.resnet50 import ResNet50
from keras.applications.resnet50 import preprocess_input

# Run input data through the ResNet50.
resnet = ResNet50(weights=’imagenet’, include_top=False)
x_train = resnet.predict(preprocess_input(x_train))
x_test = resnet.predict(preprocess_input(x_test))

from keras.models import Sequential
from keras.layers import Dense, Flatten

# Create a very simple model architecture to be trained on the
# output of ResNet50.
model = Sequential()
model.add(Flatten(input_shape=x_train.shape[1:]))
model.add(Dense(num_classes, activation=’softmax’))
# Compile the model.
model.compile(loss=’categorical_crossentropy’,
              optimizer=’adam’,
              metrics=[‘accuracy’])
```


Data
Transfer Learning
Model
Training
Evaluation

dataset.py

```python
import os
import numpy as np
from keras.preprocessing.image import load_img, img_to_array
from keras.utils.np_utils import to_categorical
from sklearn.model_selection import train_test_split

class ImageDataset:
  def __init__(self):
    self.images = []
    self.labels = []

  def add(self, dir, label):
    “”” Adds the paths to all .jpg files in the given directory
    to the dataset and labels them. Note: This does not load any
    files. “””
    files = [os.path.join(dir, f)
             for f in os.listdir(dir) if f.endswith(‘.jpg’)]
    self.images += files
    self.labels += [label] * len(files)

  def generate(self, test_split=0.33):
    “”” Loads the image files for the stored paths and returns a
    training/test split of inputs and labels in a format suitable
    for Keras. “””
    x = np.vstack([self._load_img(img) for img in self.images])
    y = to_categorical(self.labels)
    return train_test_split(x, y, test_size=test_split, random_state=42)

  def _load_img(self, path):
    “”” Loads an image from path and converts it to a 4-dimensional
    tensor to be passed to a Keras model. “””
    img = img_to_array(load_img(path, target_size=(224, 224)))
    return np.expand_dims(img, axis=0)
```

main.py

```python
# Create a very simple model architecture to be trained on the
# output of ResNet50.
model = Sequential()
model.add(Flatten(input_shape=x_train.shape[1:]))
model.add(Dense(num_classes, activation=’softmax’))
# Compile the model.
model.compile(loss=’categorical_crossentropy’,
              optimizer=’adam’,
              metrics=[‘accuracy’])

from keras.callbacks import ModelCheckpoint

# Train the model and store the weights that scored best on the
# validation set.
checkpoint = ModelCheckpoint(filepath=’best.weights.hdf5’,
                             save_best_only=True)
progress = model.fit(x_train, y_train,
                     epochs=500,
                     batch_size=75,
                     validation_split=0.33,
                     callbacks=[checkpoint])
# Restore the best weights.
model.load_weights(‘best.weights.hdf5’)
```


Data
Transfer Learning
Model
Training
Evaluation

dataset.py

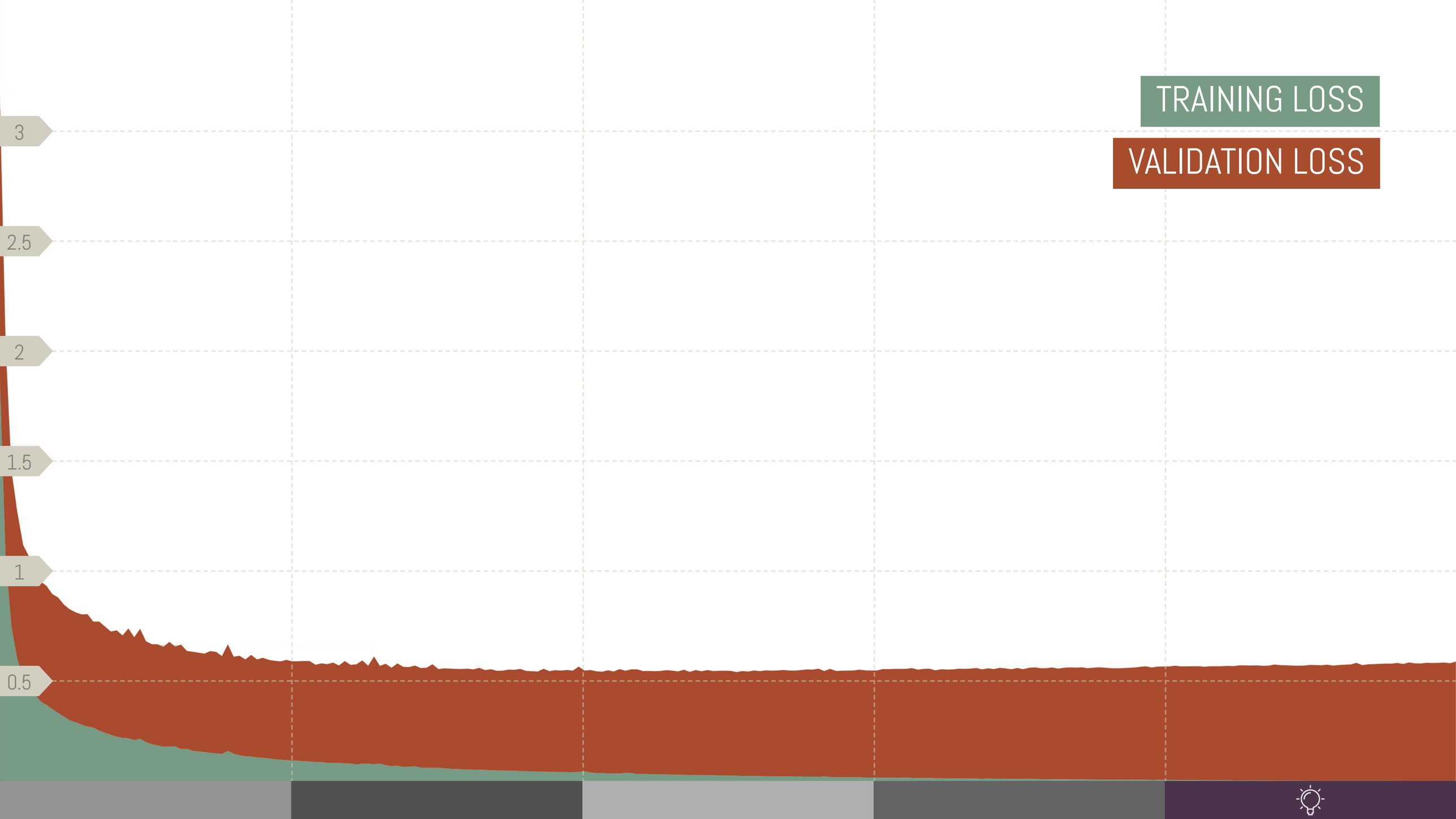
```python
import os
import numpy as np
from keras.preprocessing.image import load_img, img_to_array
from keras.utils.np_utils import to_categorical
from sklearn.model_selection import train_test_split

class ImageDataset:
  def __init__(self):
    self.images = []
    self.labels = []

  def add(self, dir, label):
    “”” Adds the paths to all .jpg files in the given directory
    to the dataset and labels them. Note: This does not load any
    files. “””
    files = [os.path.join(dir, f)
             for f in os.listdir(dir) if f.endswith(‘.jpg’)]
    self.images += files
    self.labels += [label] * len(files)

  def generate(self, test_split=0.33):
    “”” Loads the image files for the stored paths and returns a
    training/test split of inputs and labels in a format suitable
    for Keras. “””
    x = np.vstack([self._load_img(img) for img in self.images])
    y = to_categorical(self.labels)
    return train_test_split(x, y, test_size=test_split, random_state=42)

  def _load_img(self, path):
    “”” Loads an image from path and converts it to a 4-dimensional
    tensor to be passed to a Keras model. “””
    img = img_to_array(load_img(path, target_size=(224, 224)))
    return np.expand_dims(img, axis=0)
```

main.py

```python
# Create a very simple model architecture to be trained on the
# output of ResNet50.
model = Sequential()
model.add(Flatten(input_shape=x_train.shape[1:]))
model.add(Dense(num_classes, activation=’softmax’))
# Compile the model.
model.compile(loss=’categorical_crossentropy’,
              optimizer=’adam’,
              metrics=[‘accuracy’])

from keras.callbacks import ModelCheckpoint

# Train the model and store the weights that scored best on the
# validation set.
checkpoint = ModelCheckpoint(filepath=’best.weights.hdf5’,
                             save_best_only=True)
progress = model.fit(x_train, y_train,
                     epochs=500,
                     batch_size=75,
                     validation_split=0.33,
                     callbacks=[checkpoint])
# Restore the best weights.
model.load_weights(‘best.weights.hdf5’)

import numpy as np
import matplotlib.pyplot as plt

# Print best validation accuracy.
print(np.max(progress.history[‘val_acc’]))
# Plot the training and validation loss.
plt.plot(progress.history[‘loss’])
plt.plot(progress.history[‘val_loss’])
plt.xlabel(‘Epoch’)
plt.ylabel(‘Loss’)
plt.title(‘Training and Validation Loss’)
leg = plt.legend([‘Training Loss’, ‘Validation Loss’])
plt.show()
```

85.02%

VALIDATION ACCURACY


TRAINING LOSS
VALIDATION LOSS
3
2.5
2
1.5
1
0.5

dataset.py

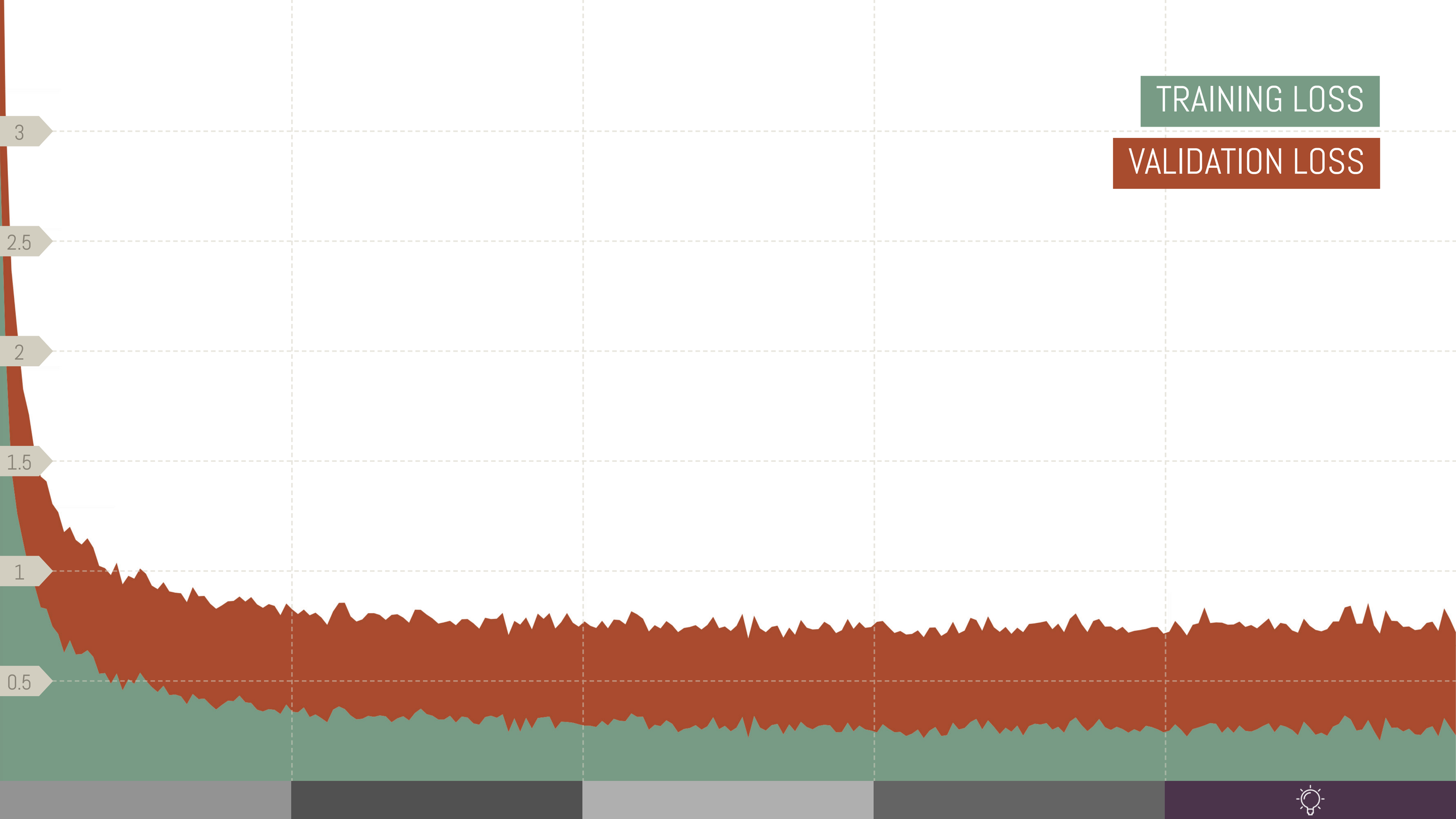
```python
import os
import numpy as np
from keras.preprocessing.image import load_img, img_to_array
from keras.utils.np_utils import to_categorical
from sklearn.model_selection import train_test_split

class ImageDataset:
  def __init__(self):
    self.images = []
    self.labels = []

  def add(self, dir, label):
    “”” Adds the paths to all .jpg files in the given directory
    to the dataset and labels them. Note: This does not load any
    files. “””
    files = [os.path.join(dir, f)
             for f in os.listdir(dir) if f.endswith(‘.jpg’)]
    self.images += files
    self.labels += [label] * len(files)

  def generate(self, test_split=0.33):
    “”” Loads the image files for the stored paths and returns a
    training/test split of inputs and labels in a format suitable
    for Keras. “””
    x = np.vstack([self._load_img(img) for img in self.images])
    y = to_categorical(self.labels)
    return train_test_split(x, y, test_size=test_split, random_state=42)

  def _load_img(self, path):
    “”” Loads an image from path and converts it to a 4-dimensional
    tensor to be passed to a Keras model. “””
    img = img_to_array(load_img(path, target_size=(224, 224)))
    return np.expand_dims(img, axis=0)
```

main.py

```python
# Create a very simple model architecture to be trained on the
# output of ResNet50.
model = Sequential()
model.add(Flatten(input_shape=x_train.shape[1:]))
model.add(Dropout(0.75))
model.add(Dense(num_classes, activation=’softmax’))
# Compile the model.
model.compile(loss=’categorical_crossentropy’,
              optimizer=’adam’,
              metrics=[‘accuracy’])

from keras.callbacks import ModelCheckpoint

# Train the model and store the weights that scored best on the
# validation set.
checkpoint = ModelCheckpoint(filepath=’best.weights.hdf5’,
                             save_best_only=True)
progress = model.fit(x_train, y_train,
                     epochs=500,
                     batch_size=75,
                     validation_split=0.33,
                     callbacks=[checkpoint])
# Restore the best weights.
model.load_weights(‘best.weights.hdf5’)

import numpy as np
import matplotlib.pyplot as plt

# Print best validation accuracy.
print(np.max(progress.history[‘val_acc’]))
# Plot the training and validation loss.
plt.plot(progress.history[‘loss’])
plt.plot(progress.history[‘val_loss’])
plt.xlabel(‘Epoch’)
plt.ylabel(‘Loss’)
plt.title(‘Training and Validation Loss’)
leg = plt.legend([‘Training Loss’, ‘Validation Loss’])
```

85.99%

VALIDATION ACCURACY


TRAINING LOSS
VALIDATION LOSS
3
2.5
2
1.5
1
0.5

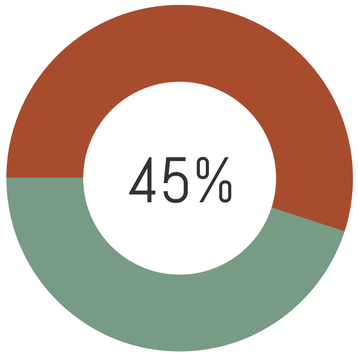
dataset.py

```python
import os
import numpy as np
from keras.preprocessing.image import load_img, img_to_array
from keras.utils.np_utils import to_categorical
from sklearn.model_selection import train_test_split

class ImageDataset:
  def __init__(self):
    self.images = []
    self.labels = []

  def add(self, dir, label):
    """ Adds the paths to all .jpg files in the given directory
    to the dataset and labels them. Note: This does not load any
    files. """
    files = [os.path.join(dir, f)
             for f in os.listdir(dir) if f.endswith('.jpg')]
    self.images += files
    self.labels += [label] * len(files)

  def generate(self, test_split=0.33):
    """ Loads the image files for the stored paths and returns a
    training/test split of inputs and labels in a format suitable
    for Keras. """
    x = np.vstack([self._load_img(img) for img in self.images])
    y = to_categorical(self.labels)
    return train_test_split(x, y, test_size=test_split, random_state=42)

  def _load_img(self, path):
    """ Loads an image from path and converts it to a 4-dimensional
    tensor to be passed to a Keras model. """
    img = img_to_array(load_img(path, target_size=(224, 224)))
    return np.expand_dims(img, axis=0)
```

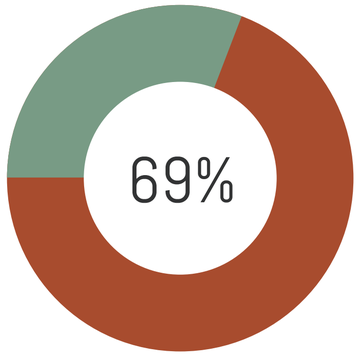
main.py

```python
# Print best validation accuracy.
print(np.max(progress.history['val_acc']))
# Plot the training and validation loss.
plt.plot(progress.history['loss'])
plt.plot(progress.history['val_loss'])
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training and Validation Loss')
leg = plt.legend(['Training Loss', 'Validation Loss'])
plt.show()

from sklearn.metrics import confusion_matrix

# Print the confusion matrix to see where most of the errors happen.
validation_split = round(0.33 * len(x_train))
predictions = model.predict_classes(x_train[-validation_split:])
expectations = np.argmax(y_train[-validation_split:], axis=1)
matrix = confusion_matrix(expectations, predictions)
print(matrix)
```

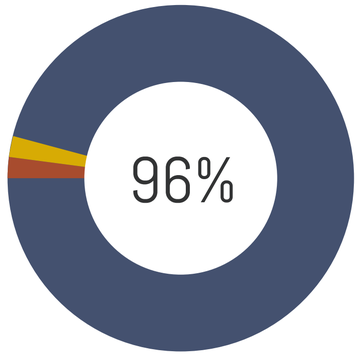
CONFUSION



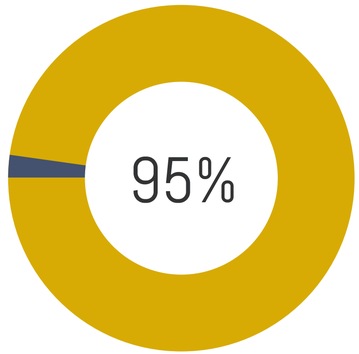
LEFT



RIGHT



FRONT



REAR

dataset.py

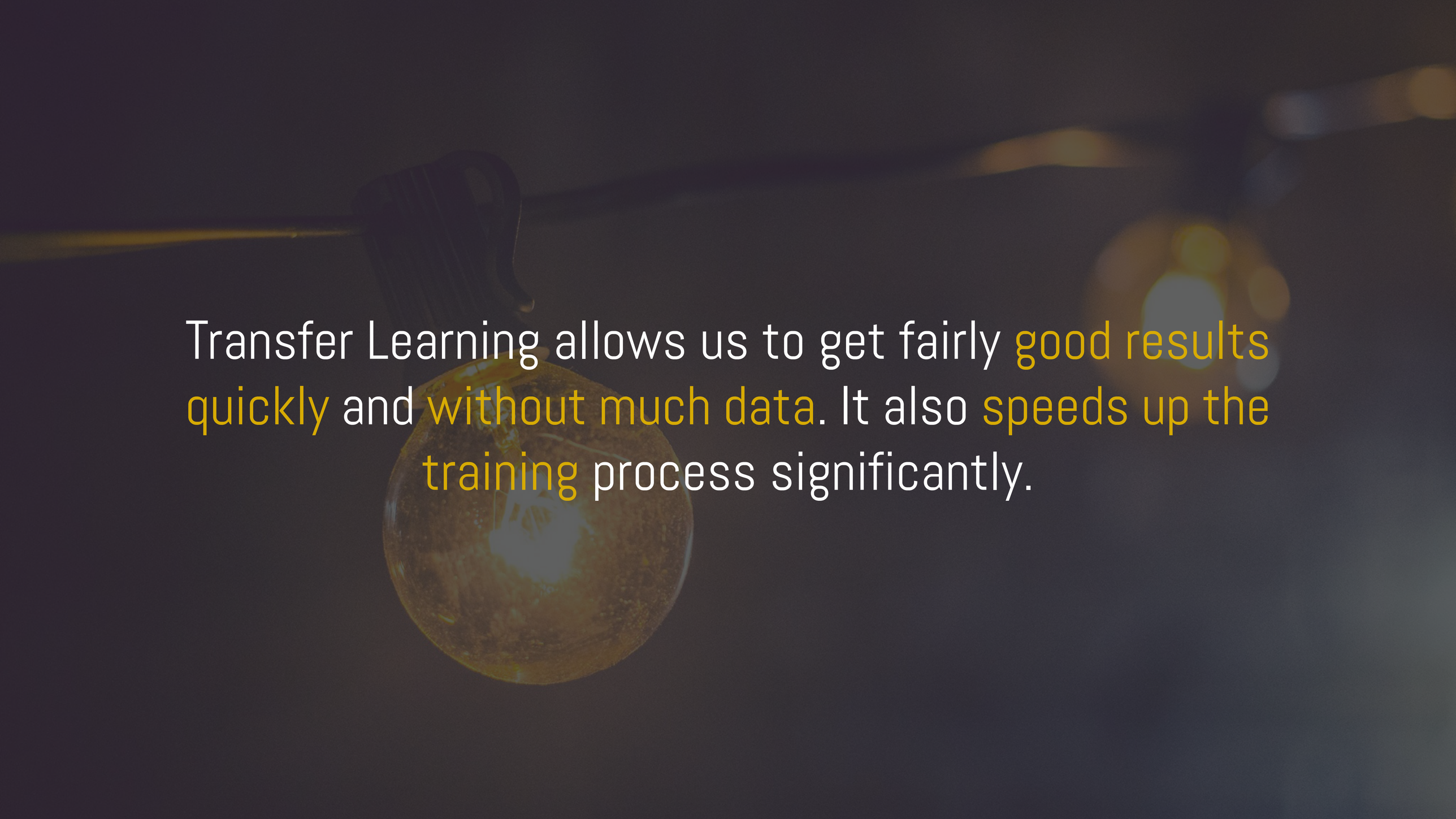
```python
import os
import numpy as np
from keras.preprocessing.image import load_img, img_to_array
from keras.utils.np_utils import to_categorical
from sklearn.model_selection import train_test_split

class ImageDataset:
  def __init__(self):
    self.images = []
    self.labels = []

  def add(self, dir, label):
    “”” Adds the paths to all .jpg files in the given directory
    to the dataset and labels them. Note: This does not load any
    files. “””
    files = [os.path.join(dir, f)
             for f in os.listdir(dir) if f.endswith(‘.jpg’)]
    self.images += files
    self.labels += [label] * len(files)

  def generate(self, test_split=0.33):
    “”” Loads the image files for the stored paths and returns a
    training/test split of inputs and labels in a format suitable
    for Keras. “””
    x = np.vstack([self._load_img(img) for img in self.images])
    y = to_categorical(self.labels)
    return train_test_split(x, y, test_size=test_split, random_state=42)

  def _load_img(self, path):
    “”” Loads an image from path and converts it to a 4-dimensional
    tensor to be passed to a Keras model. “””
    img = img_to_array(load_img(path, target_size=(224, 224)))
    return np.expand_dims(img, axis=0)
```

main.py

```python
# Print best validation accuracy.
print(np.max(progress.history[‘val_acc’]))
# Plot the training and validation loss.
plt.plot(progress.history[‘loss’])
plt.plot(progress.history[‘val_loss’])
plt.xlabel(‘Epoch’)
plt.ylabel(‘Loss’)
plt.title(‘Training and Validation Loss’)
leg = plt.legend([‘Training Loss’, ‘Validation Loss’])
plt.show()

from sklearn.metrics import confusion_matrix

# Print the confusion matrix to see where most of the errors happen.
validation_split = round(0.33 * len(x_train))
predictions = model.predict_classes(x_train[-validation_split:])
expectations = np.argmax(y_train[-validation_split:], axis=1)
matrix = confusion_matrix(expectations, predictions)
print(matrix)

# Evaluate the model on the test set.
loss, accuracy = model.evaluate(x_test, y_test)
print(‘Test Accuracy: {}’.format(accuracy))
```

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.

JOHANNES STRICKER

JOHANNESSTRICKER/TRANSFER-LEARNING.GIT

JOHANNESSTRICKER@GMX.NET

JOHANNES-STRICKER