In [11]:

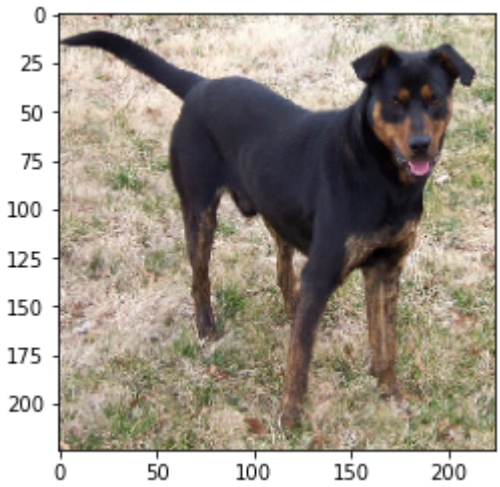
```
# example of using a pre-trained model as a classifier
from tensorflow.keras.utils import load_img
import matplotlib.pyplot as plt
# from keras.preprocessing.image import load_img
from tensorflow.keras.utils import img_to_array
from keras.applications.vgg16 import preprocess_input
from keras.applications.vgg16 import decode_predictions
from keras.applications.vgg16 import VGG16
from keras.models import Model
from pickle import dump
```

In [12]:

```
# load an image from file
image = load_img('dog.jpg', target_size=(224, 224))
plt.imshow(image)
```

Out[12]:

```
<matplotlib.image.AxesImage at 0x29742caa5e0>
```



In [13]:

```
# convert the image pixels to a numpy array
image = img_to_array(image)
# # reshape data for the model
image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
# prepare the image for the VGG model
image = preprocess_input(image)
```

In [14]:

```
# load the model
model = VGG16()
# predict the probability across all output classes
yhat = model.predict(image)
# convert the probabilities to class labels
label = decode_predictions(yhat)
# retrieve the most likely result, e.g. highest probability
label = label[0][0]
# print the classification
print('%s (%.2f%%)' % (label[1], label[2]*100))
```

```
1/1 [==============================] - 1s 1s/step
Doberman (35.42%)
```

In [5]:

```
# load an image from file
image = load_img('dog.jpg', target_size=(224, 224))
# convert the image pixels to a numpy array
image = img_to_array(image)
# reshape data for the model
image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
# prepare the image for the VGG model
image = preprocess_input(image)
# load model
model = VGG16()
# remove the output layer
model = Model(inputs=model.inputs, outputs=model.layers[-2].output)
# get extracted features
features = model.predict(image)
print(features.shape)
# save to file
dump(features, open('dog.pkl', 'wb'))
```

```
1/1 [==============================] - 3s 3s/step
(1, 4096)
```

In [ ]: