研究室ローテーション 第５回

担当：和田山 正・中井彩乃

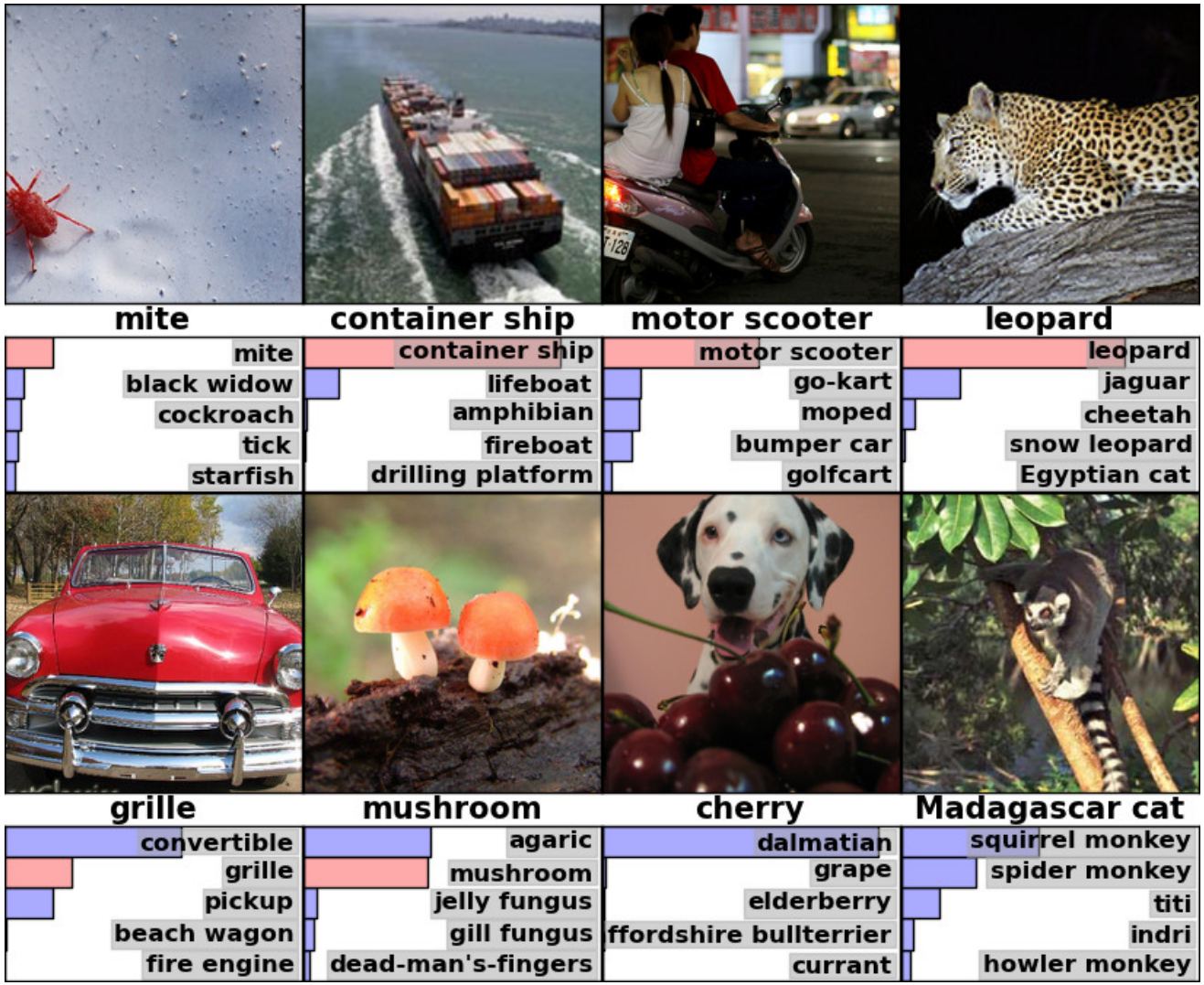
本講義の内容

- Imagenetについて
- MNIST数字認識コードを学ぶ

Imagenetにおける画像分類

- ImageNet: スタンフォード大学による画像データベース(2万カテゴリ,1400万枚の画像, ラベリングは人手)
- ILSVRC(ImageNet Large Scale Visual Recognition Challenge)を開催(公開コンペ)
- ILSVRCのデータ: 1000カテゴリ, 学習用画像120万枚, 確認用画像5万枚, テスト用画像15万枚
- 評価基準: トップ5 誤り率 (認識アルゴリズムが出力した上位の5つの答えに正解が含まれていれば認識成功)

ImageNetデータ例とTOP５出力

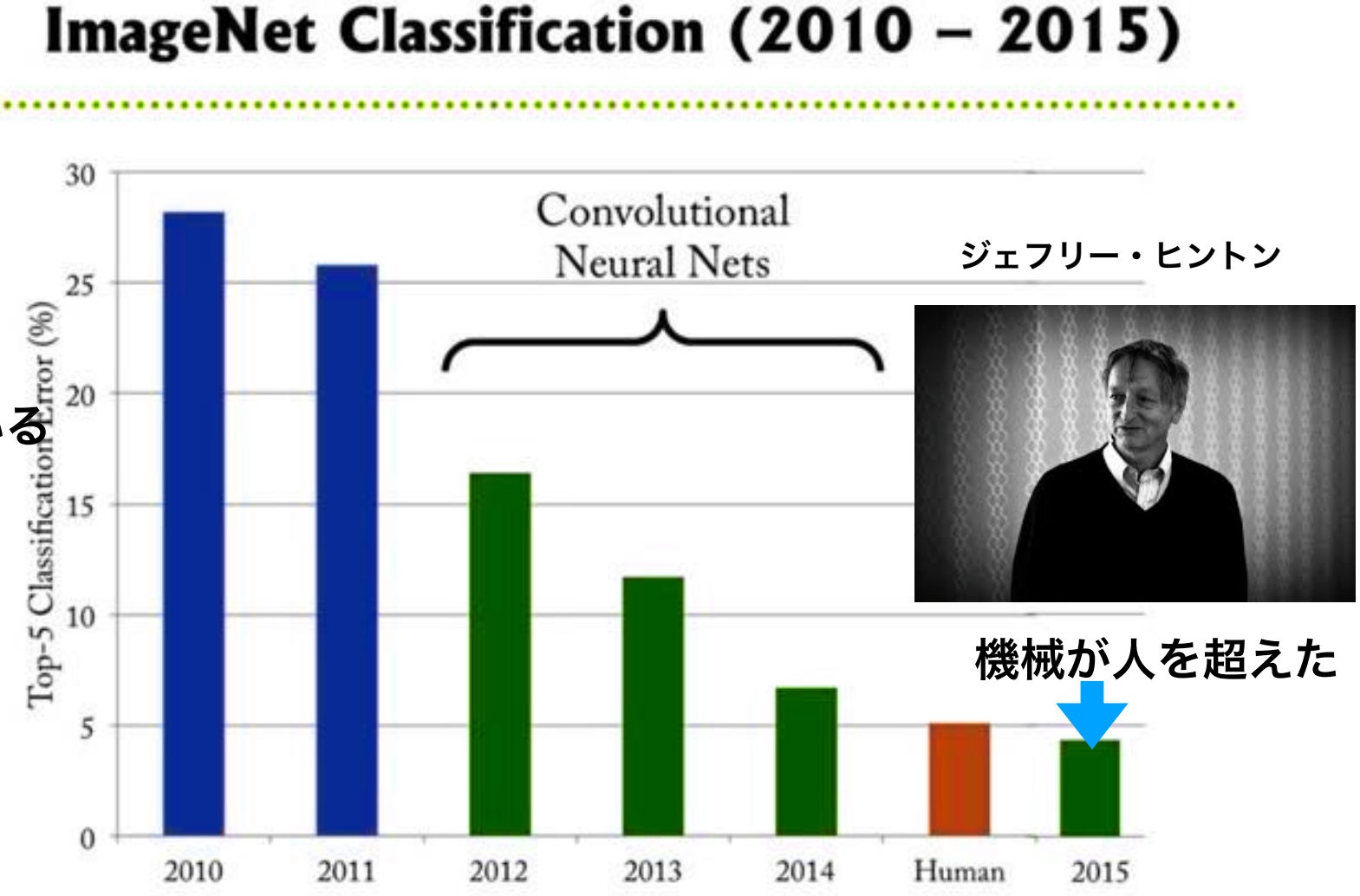


cited from: ``ImageNet Classification with Deep Convolutional Neural Networks'', Alex Krizhevsky et al.

https://www.nvidia.cn/content/tesla/pdf/machine-learning/imagenet-classification-with-deep-convolutional-nn.pdf

ImageNet認識の誤り率

性能向上に
大きく貢献している
畳込みNNは
次回の演習で
扱います



MNIST数字認識

MNIST数字認識

MNISTデータセット

- 0-9の手書き数字のデータセット
- 28 x 28ピクセル (深さ8bit, モノクロ)
- 訓練データ6万枚/テストデータ１万枚
- パターン認識アルゴリズムのテストにしばしば利用される (パターン認識の”Hello World” と呼ばれている)

MNIST数字認識コード(1)

データローダの準備 (MNISTデータのダウンロードも含む)

訓練用とテスト用でデータを分けておく

In [ ]:

```
root = '.' # mnistデータの置き場所
download = True
trans = transforms.Compose([transforms.ToTensor(), transforms.Normalize((0.5,)
, (1.0,))])
train_set = datasets.MNIST(root=root, train=True, transform=trans, download=do
wnload)
test_set = datasets.MNIST(root=root, train=False, transform=trans)
# ローダの準備
train_loader = torch.utils.data.DataLoader(dataset=train_set, batch_size=batch
_size, shuffle=True)
test_loader = torch.utils.data.DataLoader(dataset=test_set, batch_size=batch_s
ize, shuffle=False)
```

MNIST数字認識コード(2)

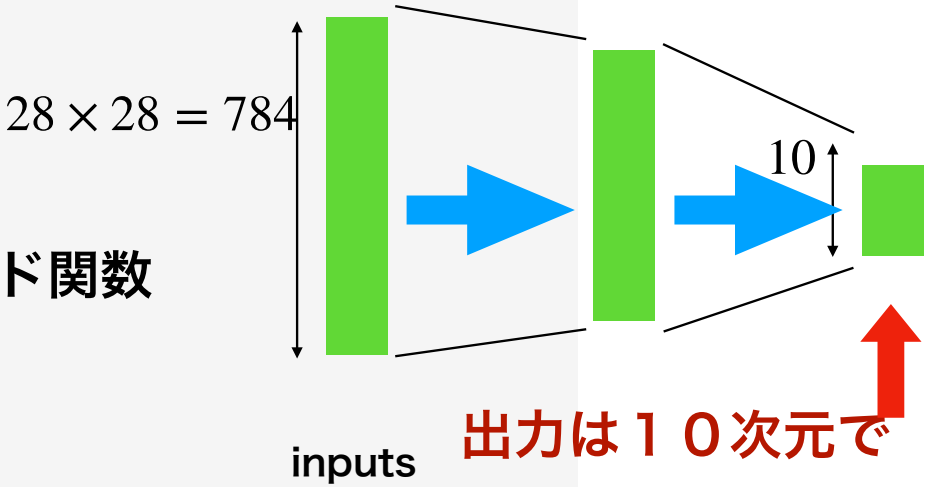
NNモデルの準備(全結合モデル)

今回は3層

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.l1 = nn.Linear(784, 32) # 28 x 28 = 784 次元の入力
        self.l2 = nn.Linear(32, 16)
        self.l3 = nn.Linear(16, 10)

    def forward(self, x):
        x = torch.sigmoid(self.l1(x))
        x = torch.sigmoid(self.l2(x))
        x = self.l3(x)
        return F.log_softmax(x, dim=1)
```

シグモイド関数



出力は１０次元で
0-9の各数字の
可能性（確率）に対応

MNIST数字認識コード(3)

学習プロセス

訓練ループは
AND関数学習の
場合とほとんど同じ

```
model = Net() # モデルのインスタンス生成
optimizer = optim.SGD(model.parameters(), lr=sgd_lr)
running_loss = 0.0
i = 0
for loop in range(3): # 3エポックの訓練
    for (input, target) in train_loader:
        i = i + 1
        input = input.view(-1, 28*28) # テンソルのサイズを整える
        optimizer.zero_grad()    # optimizerの初期化
        output = model(input)     # 推論計算
        loss = F.nll_loss(output, target) # 損失関数の定義
        loss.backward()              # バックプロパゲーション(後ろ向き計算)
        optimizer.step()             # パラメータ更新
        running_loss += loss.item()
        if i % 100 == 99:    # print every 100 mini-batches
            print('[%5d] loss: %.3f' %
                  (i + 1, running_loss / 100))
            running_loss = 0.0
```

MNIST数字認識コード(4)

推定精度(正解率)の評価

```
correct =  0 # 正解数
count = 0 # 試行数
with torch.no_grad():
    for (input, target) in test_loader:
        input = input.view(-1, 28*28)
        output = model(input)      # 推論計算
        pred = output.argmax(dim=1)
        correct += pred.eq(target.data).sum()
        count += batch_size
print ('accuracy = ', float(correct)/float(count)) # 正解率の表示
```

テストデータを利用することに注意

2次元テンソルに変換

最大値を与える要素インデックス

本講義のまとめ

- Imagenetについて
- MNIST数字認識コードを学ぶ