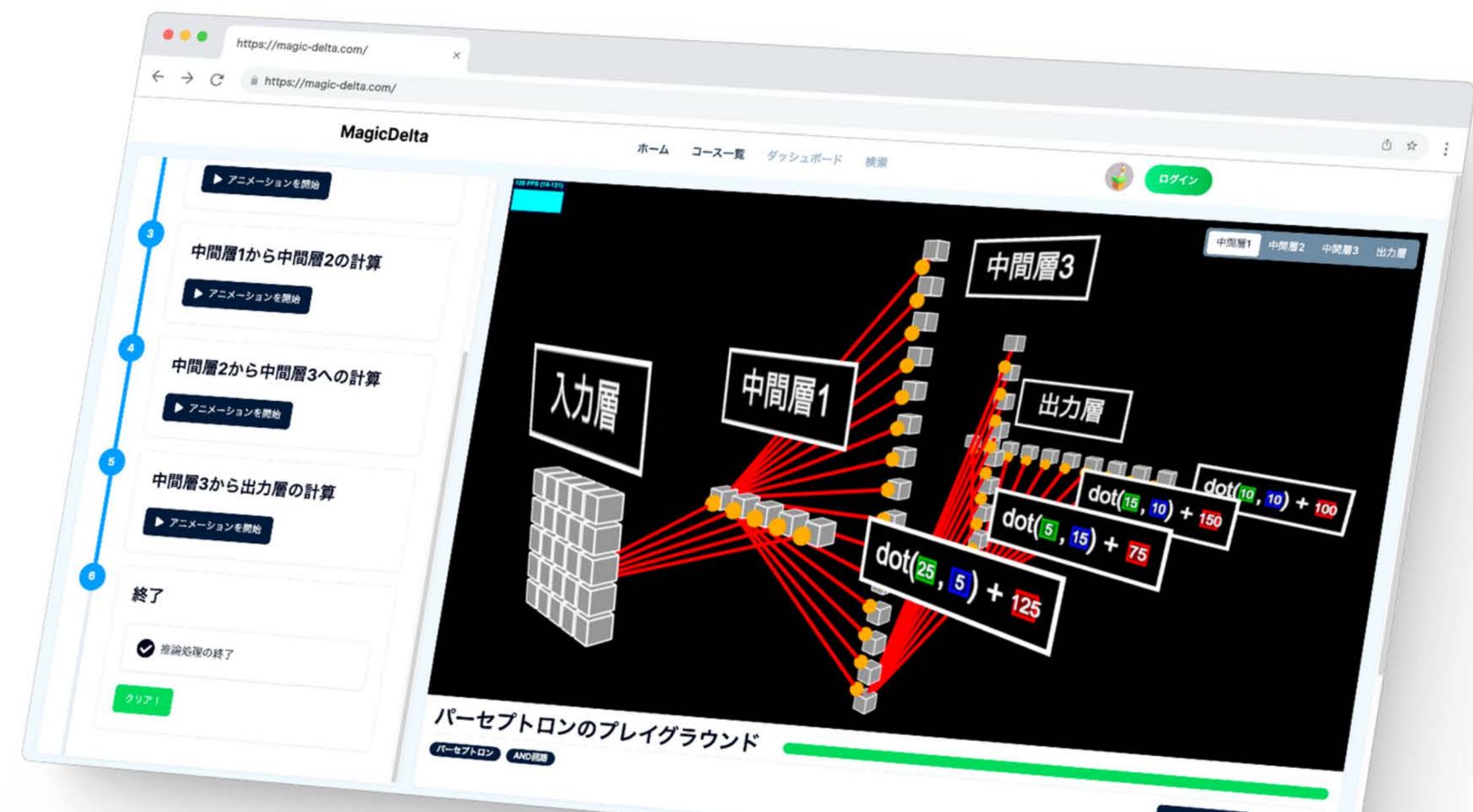




AIの仕組みを中学生でも理解できる Webアプリの開発



秋穂 正斗

目次

- ・自己紹介
- ・概要
- ・ストーリー
- ・デモ
- ・まとめ

秋穂 正斗（手羽先）

- 年齢：19歳
- 住所：福岡県 博多
- 所属：しくみデザイン株式会社
- 趣味：カラオケ, ボカロ, 日常
- 特技：無限に寝る, ルービックキューブ, Scratch7年
- 分野：AI, VPL, Game, Web



手羽先 🎉 @EDD @福岡未踏 @Tebasaki_lab · 5月26日
自作ビジュアルプログラミング言語、アゴが外れる問題は解決したのに今度
はブロックが逃げるようになって困ってる



15

896

5,971

8

113

1,111

43万



バックグラウンド



- 小学5年
 - Scratchを始める
- 高校1年
 - プログラミング
- 高校2年
 - AI
 - 大学数学を独学
- 今年3月にN高を卒業

高校2年生をAIに溶かした

開発駆動コース



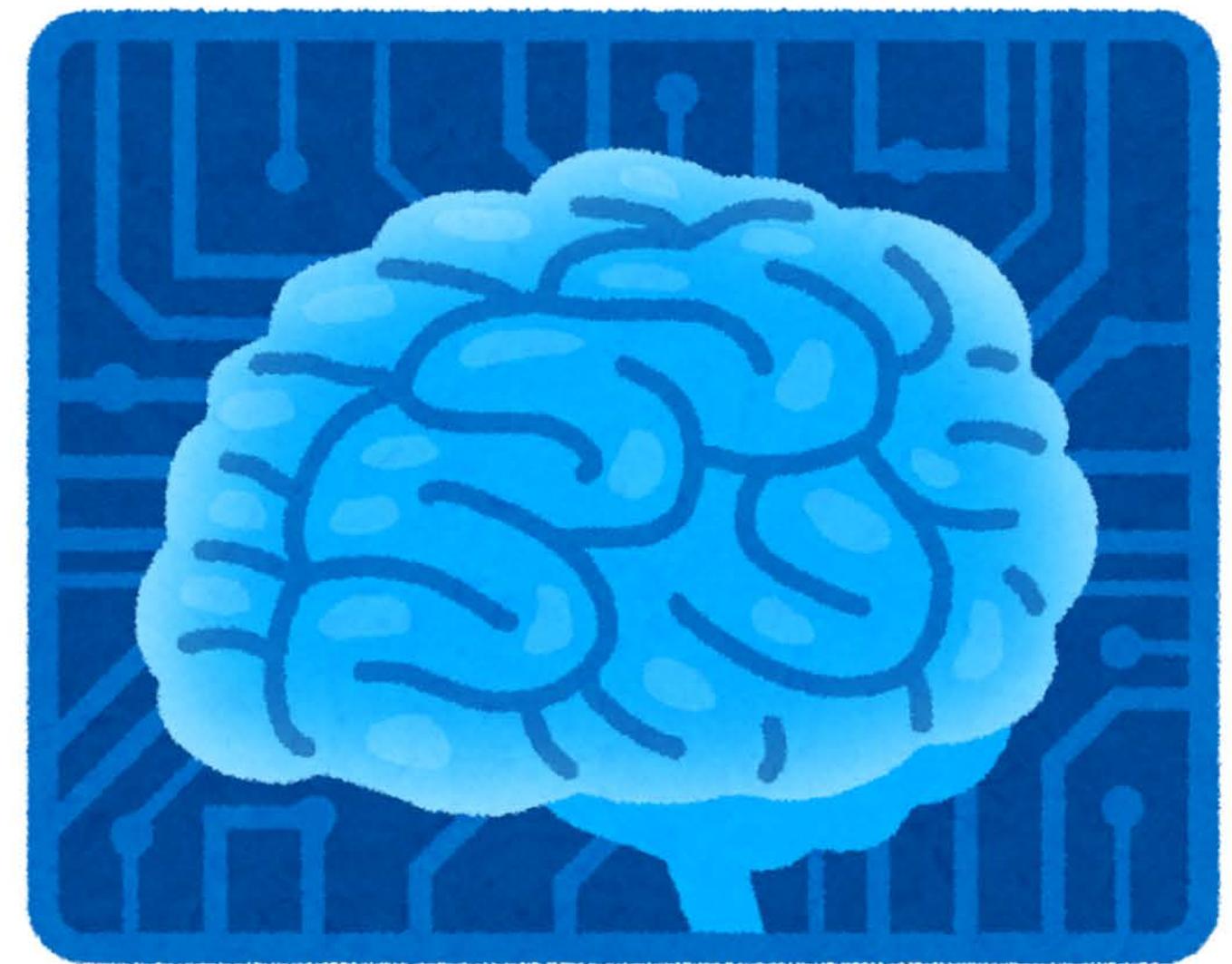
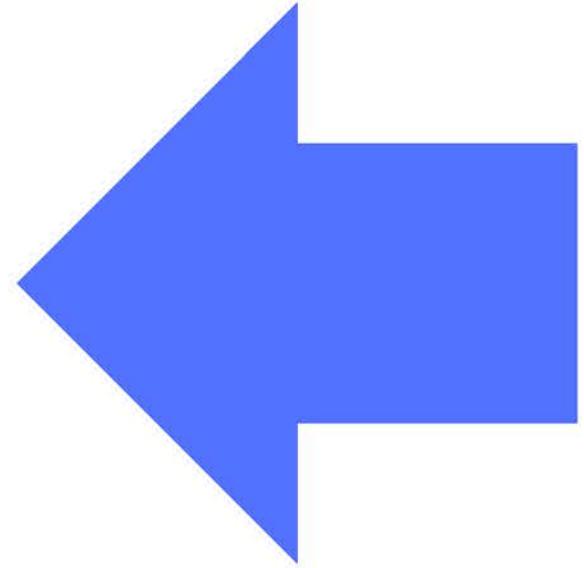
- まず開発をしてみて、作りたいものを決める
 - 開発が原動→開発駆動
- 実装の割合が多い
 - 考える、まとめるよりもゴリゴリ開発するイメージ

作りたいもの



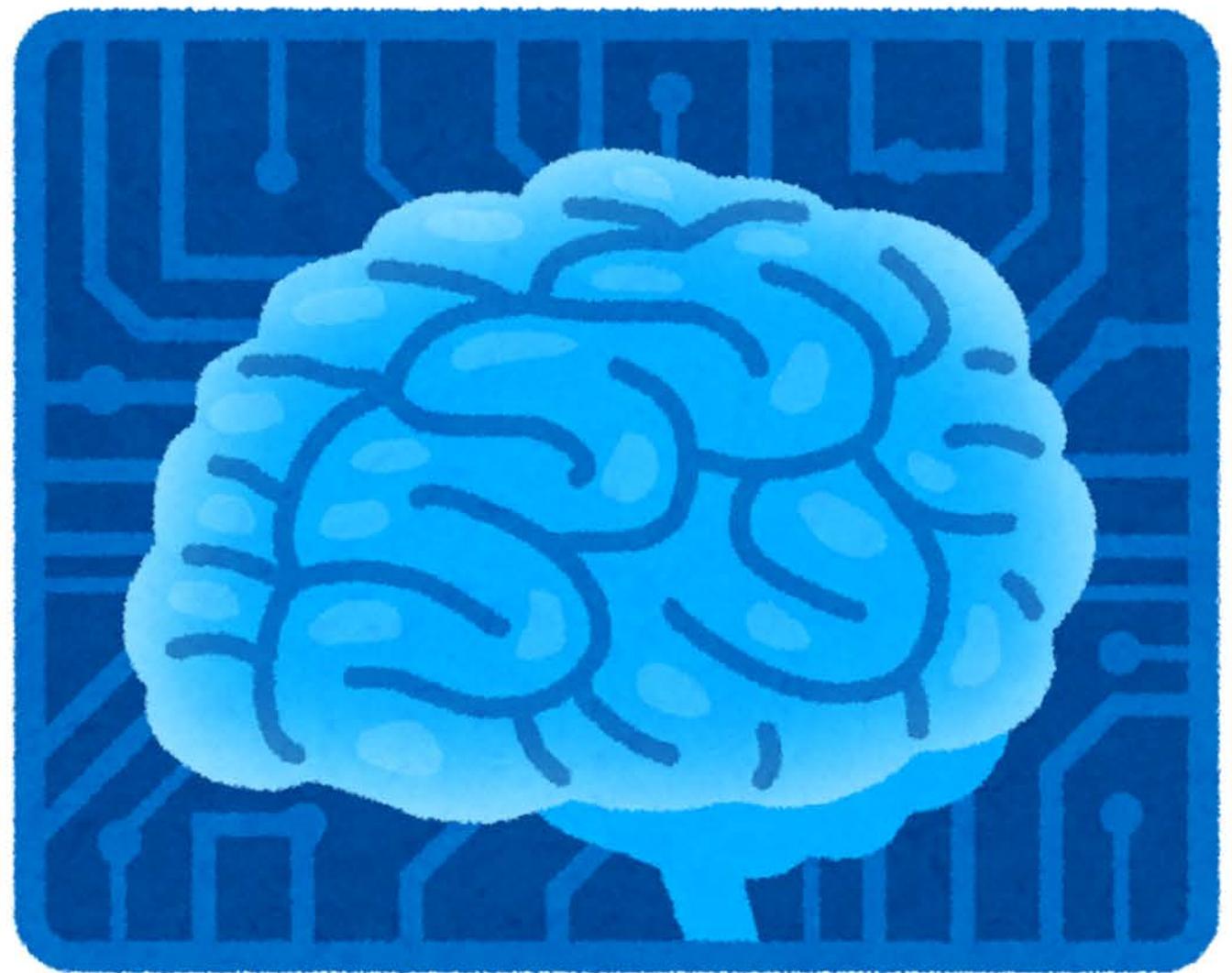
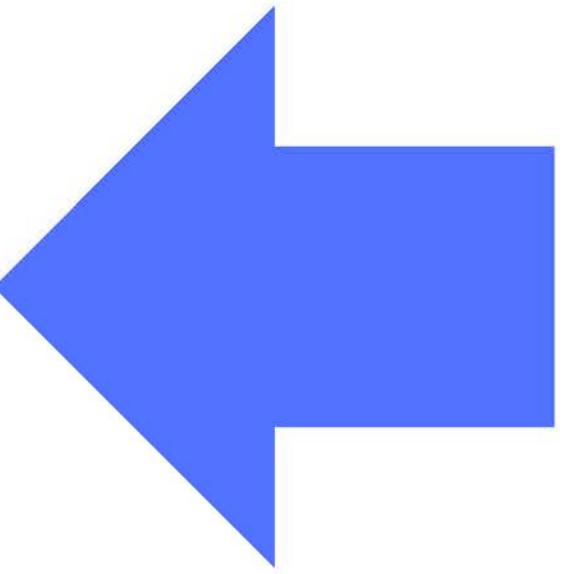
- AIの脆弱性を理解できるWebサービス
- ゲーム感覚で学習 / 実験できる

画像認識をするAI

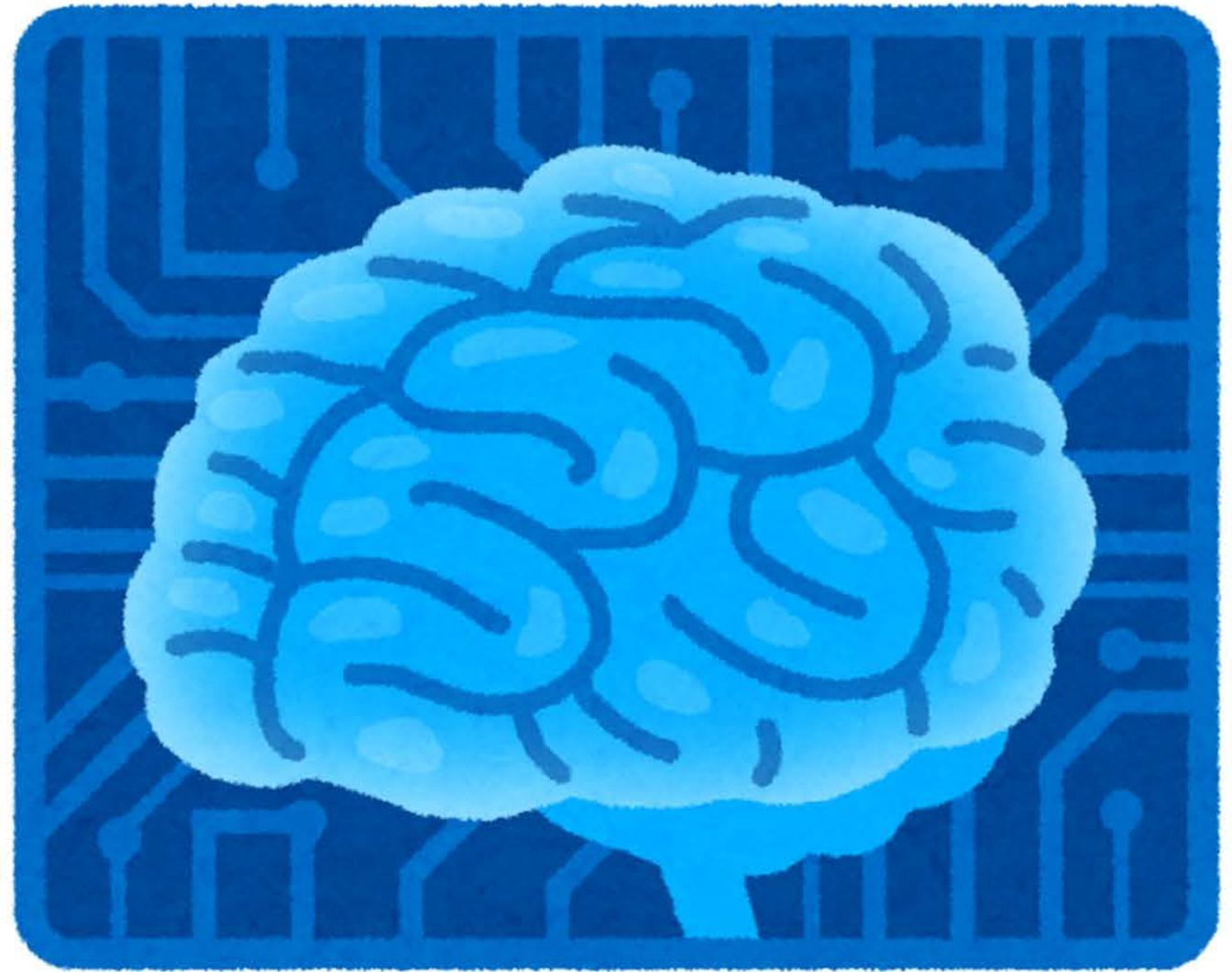
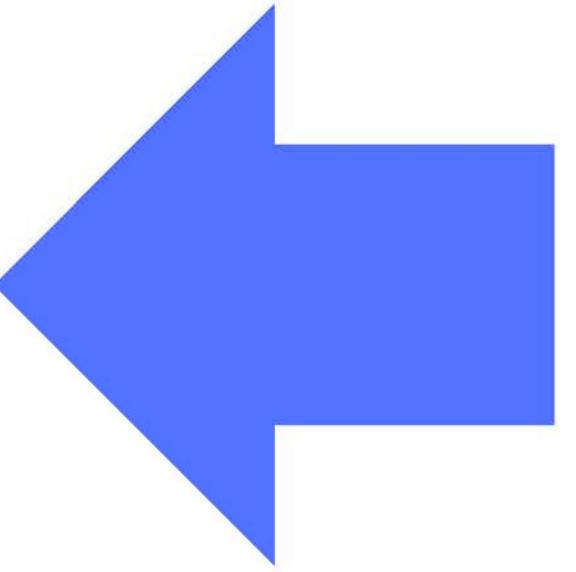


[2] Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy. "Explaining and Harnessing Adversarial Examples." arXiv:1412.6572

テナガザル！

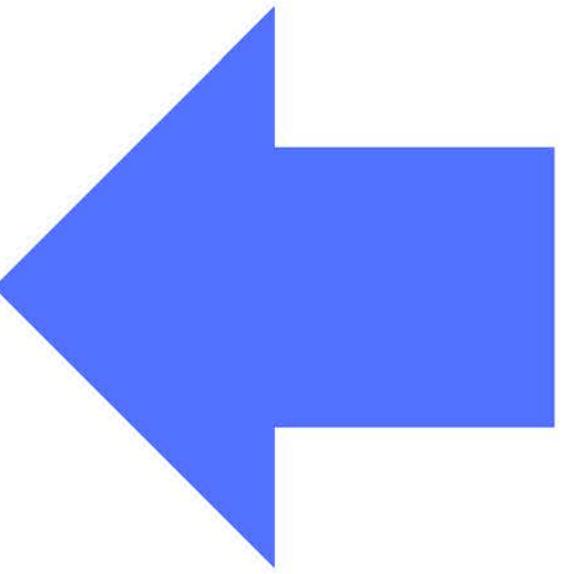


パンダ！

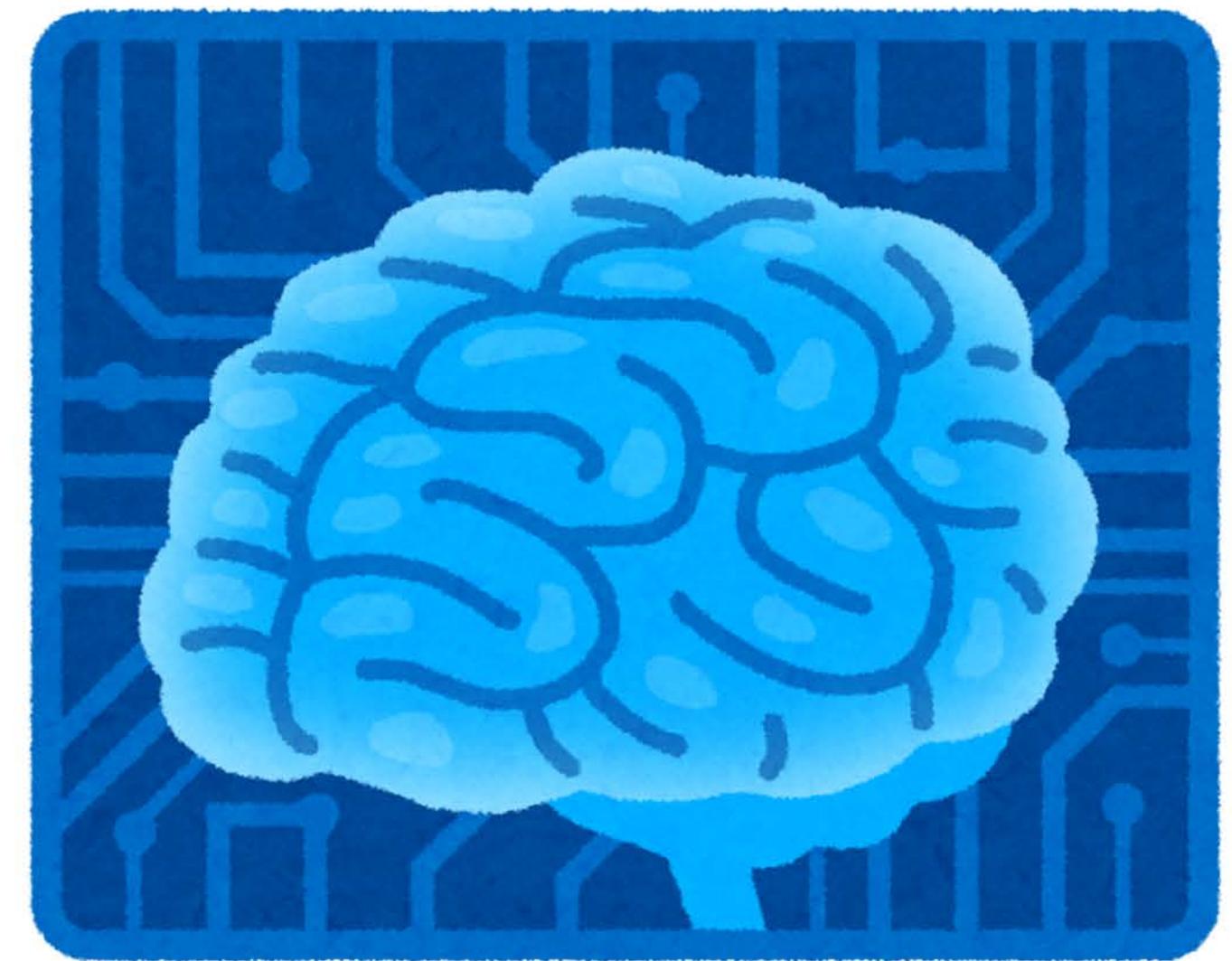




+



テナガザル！



[2] Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy. "Explaining and Harnessing Adversarial Examples." arXiv:1412.6572

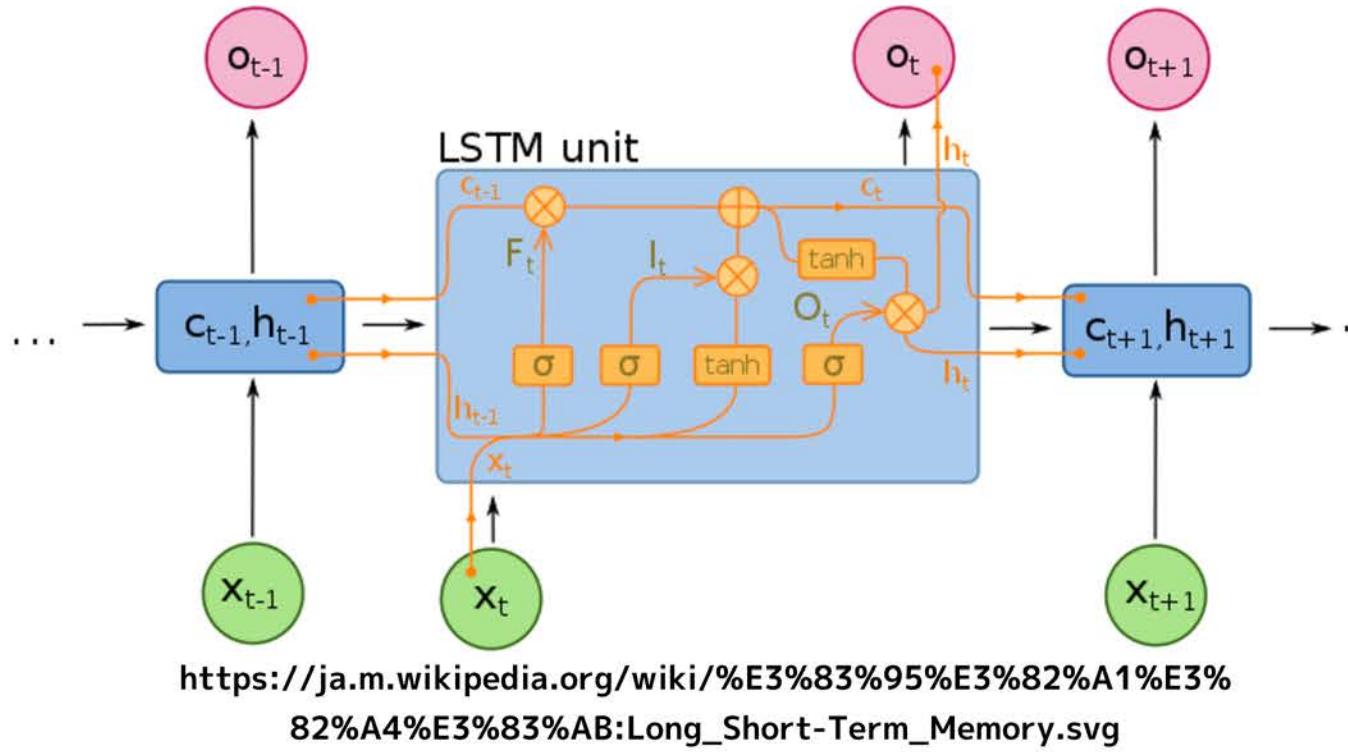
AIの脆弱性の危険性

- 画像認識
 - 自動運転の標識
- 音声認識
 - 間違った命令
- テキスト
 - バイアスのある発言



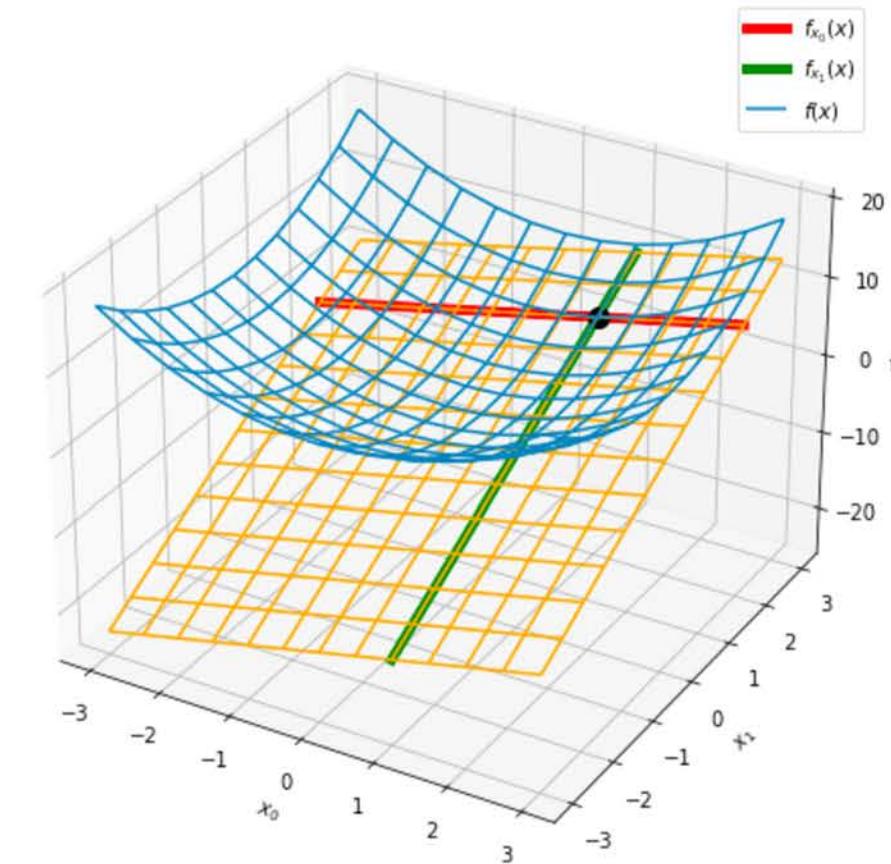
- ・ビジョンや理念
- ・ターゲット

AIの仕組みは難しい



$$f(x) = x_0^2 + x_1^2$$

$(x_0, x_1, f(x)) = (1.0, 2.0, 5.0), (dx_0, dx_1) = (2.0, 4.0)$



```

class SimpleCNN(nn.Module):
    def __init__(self):
        super(SimpleCNN, self).__init__()
        self.layer1 = nn.Sequential(
            nn.Conv2d(1, 32, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2)
        )
        self.layer2 = nn.Sequential(
            nn.Conv2d(32, 64, kernel_size=3),
            nn.ReLU(),
            nn.MaxPool2d(2)
        )
        self.fc1 = nn.Linear(64 * 6 * 6, 1000)
        self.fc2 = nn.Linear(1000, 10)

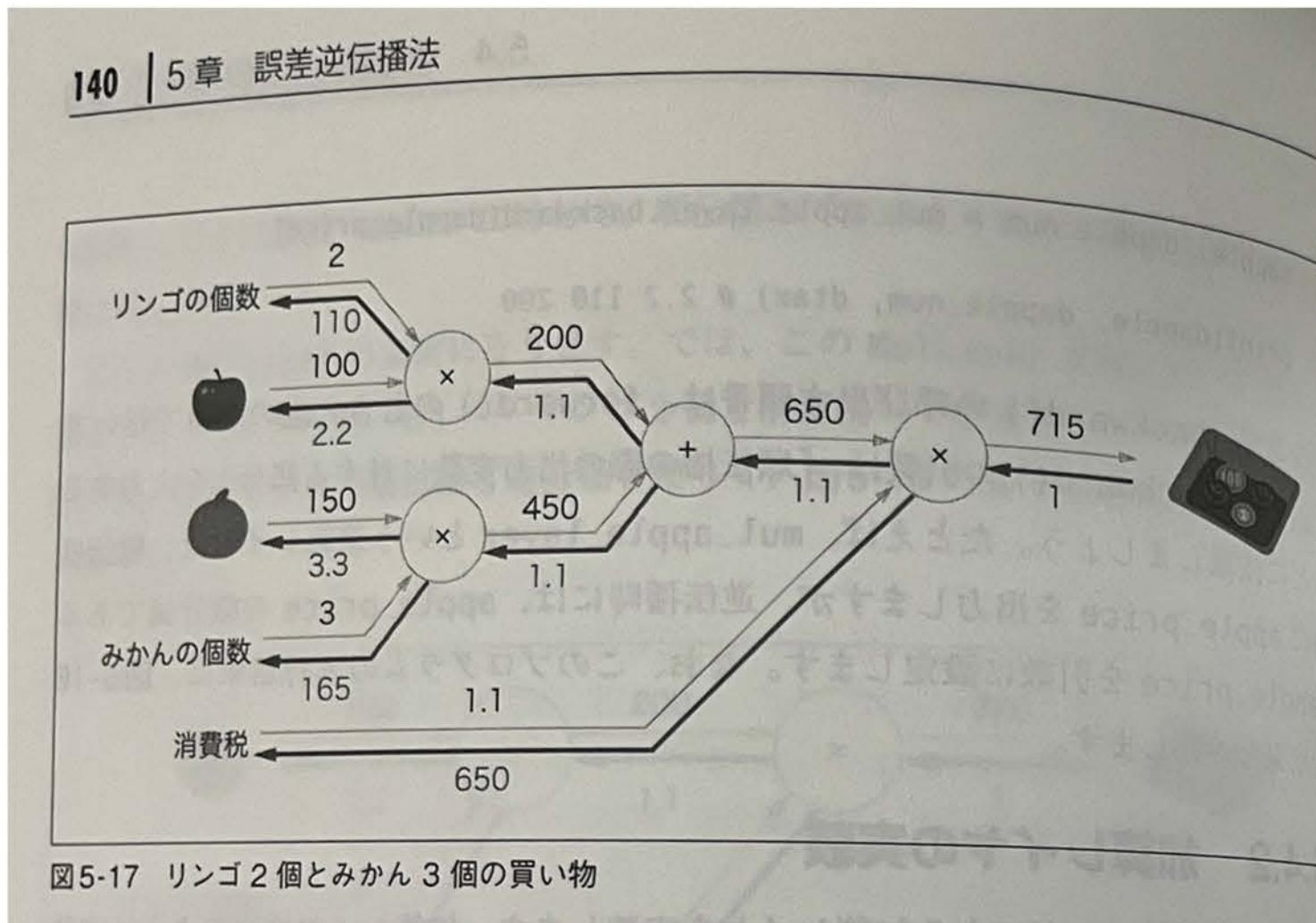
    def forward(self, x):
        out = self.layer1(x)
        out = self.layer2(out)
        out = out.view(out.size(0), -1)
        out = self.fc1(out)
        out = self.fc2(out)
        return out

```

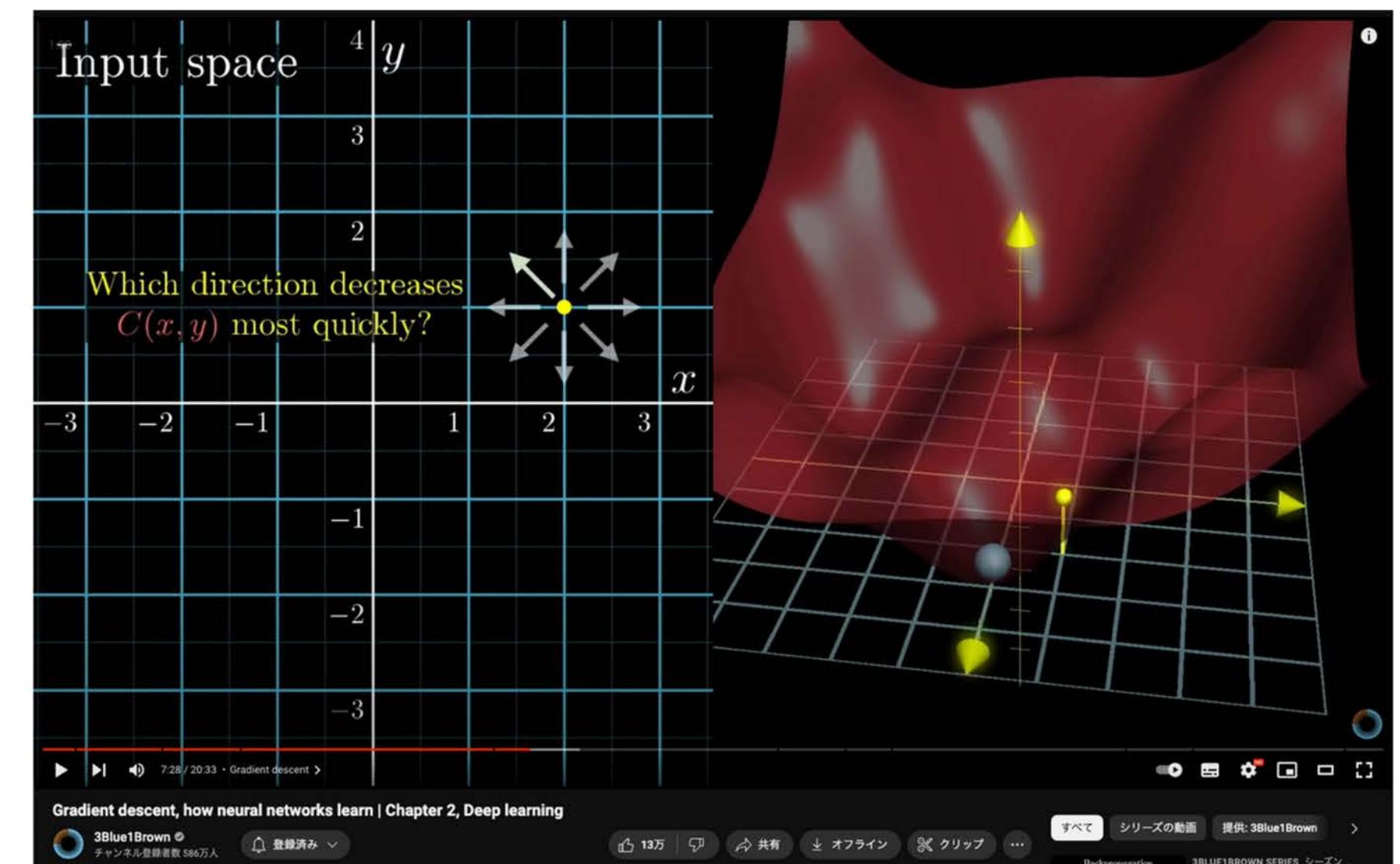
$$\begin{aligned} \frac{\partial L}{\partial w_{1,1}} &= \frac{\partial L}{\partial y_{1,1}} \frac{\partial y_{1,1}}{\partial w_{1,1}} + \frac{\partial L}{\partial y_{2,1}} \frac{\partial y_{2,1}}{\partial w_{1,1}} + \dots + \frac{\partial L}{\partial y_{N,1}} \frac{\partial y_{N,1}}{\partial w_{1,1}} \\ &= \sum_{h=1}^H \frac{\partial L}{\partial y_{n,1}} \frac{\partial y_{n,1}}{\partial w_{1,1}} \end{aligned}$$

$$\frac{\partial L}{\partial W} = \begin{pmatrix} \sum_{n=1}^N x_{n,1} & \sum_{n=1}^N x_{n,2} & \dots & \sum_{n=1}^N x_{n,D} \\ \sum_{n=1}^N x_{n,2} & \sum_{n=1}^N x_{n,2} & \dots & \sum_{n=1}^N x_{n,2} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{n=1}^N x_{n,D} & \sum_{n=1}^N x_{n,D} & \dots & \sum_{n=1}^N x_{n,D} \end{pmatrix} \begin{pmatrix} \sum_{n=1}^N \frac{\partial L}{\partial y_{n,1}} & \sum_{n=1}^N \frac{\partial L}{\partial y_{n,1}} & \dots & \sum_{n=1}^N \frac{\partial L}{\partial y_{n,1}} \\ \sum_{n=1}^N \frac{\partial L}{\partial y_{n,2}} & \sum_{n=1}^N \frac{\partial L}{\partial y_{n,2}} & \dots & \sum_{n=1}^N \frac{\partial L}{\partial y_{n,2}} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{n=1}^N \frac{\partial L}{\partial y_{n,H}} & \sum_{n=1}^N \frac{\partial L}{\partial y_{n,H}} & \dots & \sum_{n=1}^N \frac{\partial L}{\partial y_{n,H}} \end{pmatrix}$$

書籍や動画で出来ないことは？



ゼロから作るDeep Learning



→動かすことができない…

アニメーション+GUI

- 動きや仕組みを可視化する
- 可視化したものをユーザーが動かせる

アニメーションで可視化する

視点や位置を自由に動かせる

福岡未踏

ホーム コース一覧 ダッシュボード 検索 ログイン

中間層1 中間層2 中間層3 出力層

1 入力層、中間層、出力層の生成
⌚ 5×5の入力を作る
⌚ 再生中

2 入力層から中間層への計算
⌚ 前のチャプターをクリアしましょう

3 中間層1から中間層2への計算
⌚ 前のチャプターをクリアしましょう

4 中間層2から中間層3への計算
⌚ 前のチャプターをクリアしましょう

5 中間層3から出力層の計算
⌚ 前のチャプターをクリアしましょう

114 FPS (100-121)

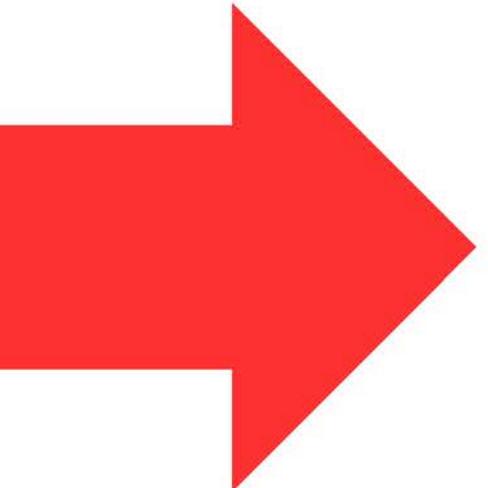
パーセプトロンのプレイグラウンド

パーセプトロン AND回路 スライドを見る ヒント

The screenshot shows a neural network visualization tool. On the left, there's a sidebar with five numbered steps: 1. 入力層、中間層、出力層の生成; 2. 入力層から中間層への計算; 3. 中間層1から中間層2への計算; 4. 中間層2から中間層3への計算; 5. 中間層3から出力層の計算. Each step has a corresponding button below it. In the center, there's a large 3D perspective diagram of a neural network with three hidden layers labeled '中間層1', '中間層2', and '中間層3', and an output layer labeled '出力層'. The diagram shows the flow of data from the input layer through the hidden layers to the output layer, with mathematical operations like 'dot' and '+' indicated. At the bottom, there's a section titled 'パーセプトロンのプレイグラウンド' with buttons for 'パーセプトロン' and 'AND回路', and links for 'スライドを見る' and 'ヒント'.

90%以上コード削減！

```
● ● ●  
import * as THREE from 'three';  
import { Tween, Easing } from '@tweenjs/tween.js';  
  
// ニューラルネットワークの構造を定義します。  
const layers = [[3], [4, 3], [2]];  
  
// ニューロンの位置を計算するための間隔  
const spacing = 2;  
  
// ニューロンを表現する球体の半径  
const neuronRadius = 0.5;  
  
// ニューロン間を接続する線の材料  
const lineMaterial = new THREE.LineBasicMaterial({ color: 0xfffff  
// シーンを作成  
const scene = new THREE.Scene();  
  
// ニューロンと接続を描画  
let neurons = [];  
layers.forEach((layer, layerIndex) => {  
  neurons[layerIndex] = [];  
  layer.forEach((rowSize, rowIndex) => {  
    neurons[layerIndex][rowIndex] = [];  
    for (let neuronIndex = 0; neuronIndex < rowSize; neuronIndex++) {  
      // ニューロンの位置を計算  
      const x = layerIndex * spacing;  
      const y = (neuronIndex - rowSize / 2) * spacing;  
      const z = (rowIndex - layer.length / 2) * spacing;  
  
      // ニューロンを表現する球体を作成  
      const neuronGeometry = new THREE.SphereGeometry(neuronRadius);  
      const neuronMaterial = new THREE.MeshBasicMaterial({ color: 0x00ff00  
      const neuron = new THREE.Mesh(neuronGeometry, neuronMaterial);  
      neuron.position.set(x, y, z);  
      scene.add(neuron);  
  
      neurons[layerIndex][rowIndex].push(neuron);  
    }  
  });  
});  
  
// アニメーションの作成  
for (let layerIndex = 0; layerIndex < layers.length - 1; layerIndex++) {  
  for (let rowIndex = 0; rowIndex < layers[layerIndex].length; rowIndex++) {  
    for (let neuronIndex = 0; neuronIndex < layers[layerIndex][rowIndex].length; neuronIndex++) {  
      const targetX = layers[layerIndex + 1][rowIndex] * spacing;  
      const targetY = (layers[layerIndex][rowIndex][neuronIndex].position.y - spacing) * spacing;  
      const targetZ = (layers[layerIndex][rowIndex].length / 2) * spacing;  
  
      const tween = new Tween(layers[layerIndex][rowIndex][neuronIndex].position).  
        to({ x: targetX, y: targetY, z: targetZ }, 1000, Easing.  
        start('onComplete', () => {  
          if (layerIndex === layers.length - 2) {  
            for (let i = 0; i < layers[layerIndex + 1].length; i++) {  
              for (let j = 0; j < layers[layerIndex + 1][i].length; j++) {  
                for (let k = 0; k < layers[layerIndex + 1][i][j].length; k++) {  
                  const targetX = layers[layerIndex + 2][i][j] * spacing;  
                  const targetY = (layers[layerIndex + 1][i][j][k].position.y - spacing) * spacing;  
                  const targetZ = (layers[layerIndex + 1][i][j].length / 2) * spacing;  
  
                  const tween = new Tween(layers[layerIndex + 1][i][j][k].position).  
                    to({ x: targetX, y: targetY, z: targetZ }, 1000, Easing.  
                    start('onComplete', () => {  
                      if (layerIndex === layers.length - 3) {  
                        for (let l = 0; l < layers[layerIndex + 2].length; l++) {  
                          for (let m = 0; m < layers[layerIndex + 2][l].length; m++) {  
                            for (let n = 0; n < layers[layerIndex + 2][l][m].length; n++) {  
                              const targetX = layers[layerIndex + 3][l][m] * spacing;  
                              const targetY = (layers[layerIndex + 2][l][m][n].position.y - spacing) * spacing;  
                              const targetZ = (layers[layerIndex + 2][l][m].length / 2) * spacing;  
  
                              const tween = new Tween(layers[layerIndex + 2][l][m][n].position).  
                                to({ x: targetX, y: targetY, z: targetZ }, 1000, Easing.  
                                start('onComplete', () => {  
                                  if (layerIndex === layers.length - 4) {  
                                    for (let o = 0; o < layers[layerIndex + 3].length; o++) {  
                                      for (let p = 0; p < layers[layerIndex + 3][o].length; p++) {  
                                        for (let q = 0; q < layers[layerIndex + 3][o][p].length; q++) {  
                                          const targetX = layers[layerIndex + 4][o][p] * spacing;  
                                          const targetY = (layers[layerIndex + 3][o][p][q].position.y - spacing) * spacing;  
                                          const targetZ = (layers[layerIndex + 3][o][p].length / 2) * spacing;  
  
                                          const tween = new Tween(layers[layerIndex + 3][o][p][q].position).  
                                            to({ x: targetX, y: targetY, z: targetZ }, 1000, Easing.  
                                            start('onComplete', () => {  
                                              if (layerIndex === layers.length - 5) {  
                                                for (let r = 0; r < layers[layerIndex + 4].length; r++) {  
                                                  for (let s = 0; s < layers[layerIndex + 4][r].length; s++) {  
                                                    for (let t = 0; t < layers[layerIndex + 4][r][s].length; t++) {  
                                                      const targetX = layers[layerIndex + 5][r][s] * spacing;  
                                                      const targetY = (layers[layerIndex + 4][r][s][t].position.y - spacing) * spacing;  
                                                      const targetZ = (layers[layerIndex + 4][r][s].length / 2) * spacing;  
  
                                                      const tween = new Tween(layers[layerIndex + 4][r][s][t].position).  
                                                        to({ x: targetX, y: targetY, z: targetZ }, 1000, Easing.  
                                                        start('onComplete', () => {  
                                                          if (layerIndex === layers.length - 6) {  
                                                            for (let u = 0; u < layers[layerIndex + 5].length; u++) {  
                                                              for (let v = 0; v < layers[layerIndex + 5][u].length; v++) {  
                                                                for (let w = 0; w < layers[layerIndex + 5][u][v].length; w++) {  
                                                                  const targetX = layers[layerIndex + 6][u][v] * spacing;  
                                                                  const targetY = (layers[layerIndex + 5][u][v][w].position.y - spacing) * spacing;  
                                                                  const targetZ = (layers[layerIndex + 5][u][v].length / 2) * spacing;  
  
                                                                  const tween = new Tween(layers[layerIndex + 5][u][v][w].position).  
                                                                    to({ x: targetX, y: targetY, z: targetZ }, 1000, Easing.  
                                                                    start('onComplete', () => {  
                                                                      if (layerIndex === layers.length - 7) {  
                                                                        for (let x = 0; x < layers[layerIndex + 6].length; x++) {  
                                                                          for (let y = 0; y < layers[layerIndex + 6][x].length; y++) {  
                                                                            for (let z = 0; z < layers[layerIndex + 6][x][y].length; z++) {  
                                                                              const targetX = layers[layerIndex + 7][x][y] * spacing;  
                                                                              const targetY = (layers[layerIndex + 6][x][y][z].position.y - spacing) * spacing;  
                                                                              const targetZ = (layers[layerIndex + 6][x][y].length / 2) * spacing;  
  
                                                                              const tween = new Tween(layers[layerIndex + 6][x][y][z].position).  
                                                                                to({ x: targetX, y: targetY, z: targetZ }, 1000, Easing.  
                                                                                start('onComplete', () => {  
                                                                                  if (layerIndex === layers.length - 8) {  
                                                                                    for (let a = 0; a < layers[layerIndex + 7].length; a++) {  
                                                                                      for (let b = 0; b < layers[layerIndex + 7][a].length; b++) {  
                        
```



```
● ● ●  
const layer = [  
  { y: 5, x: 5, gap: 0.5},  
  { y: 1, x: 5 },  
  { y: 15, x: 1 },  
  { y: 10, x: 1 },  
  { y: 1, x: 10 },  
];  
  
create3DLayer(layer);
```

実際の10行のコード

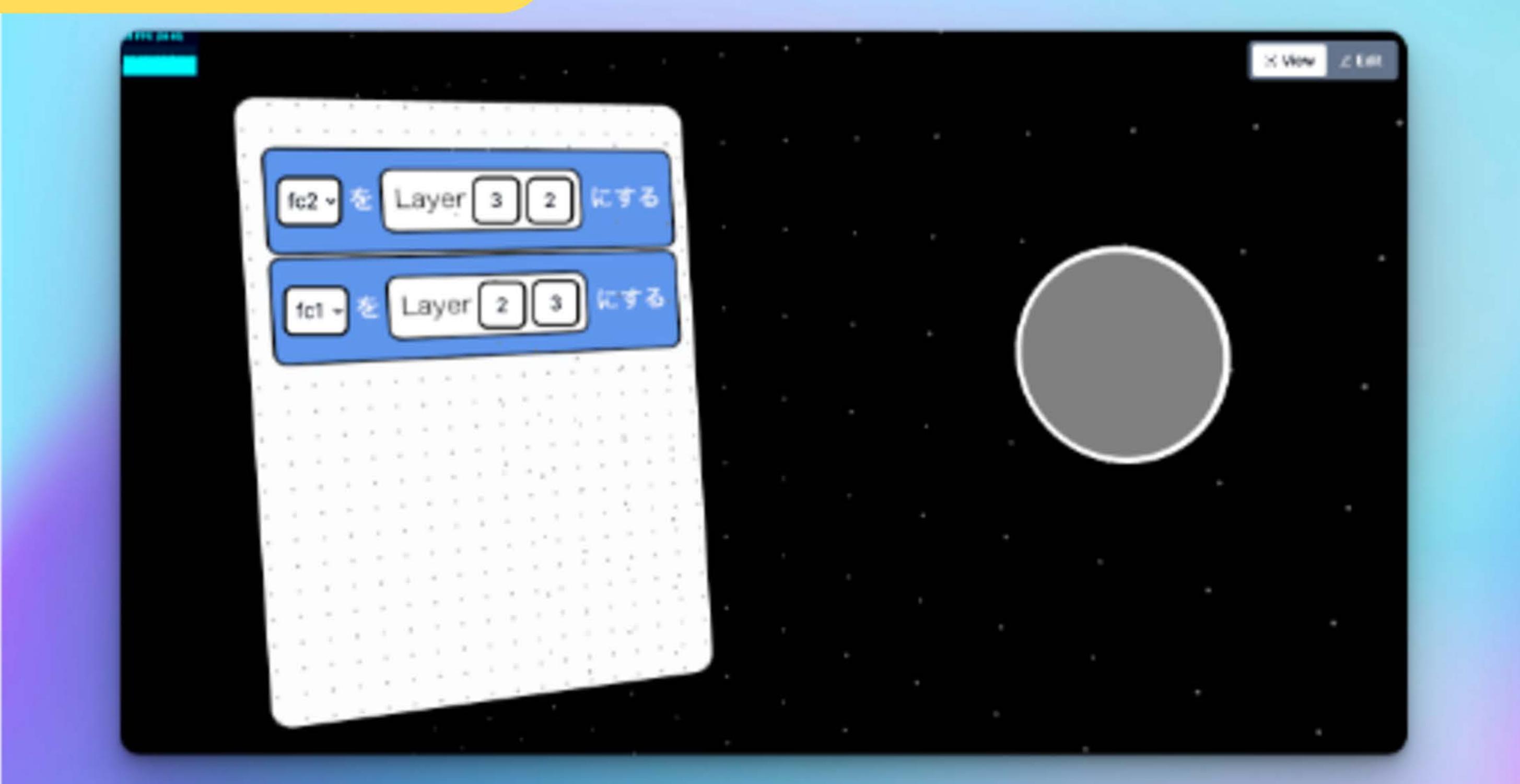
約100~500行のコード

直感的に理解できる

スライダーなどで数式を直感的に変更！

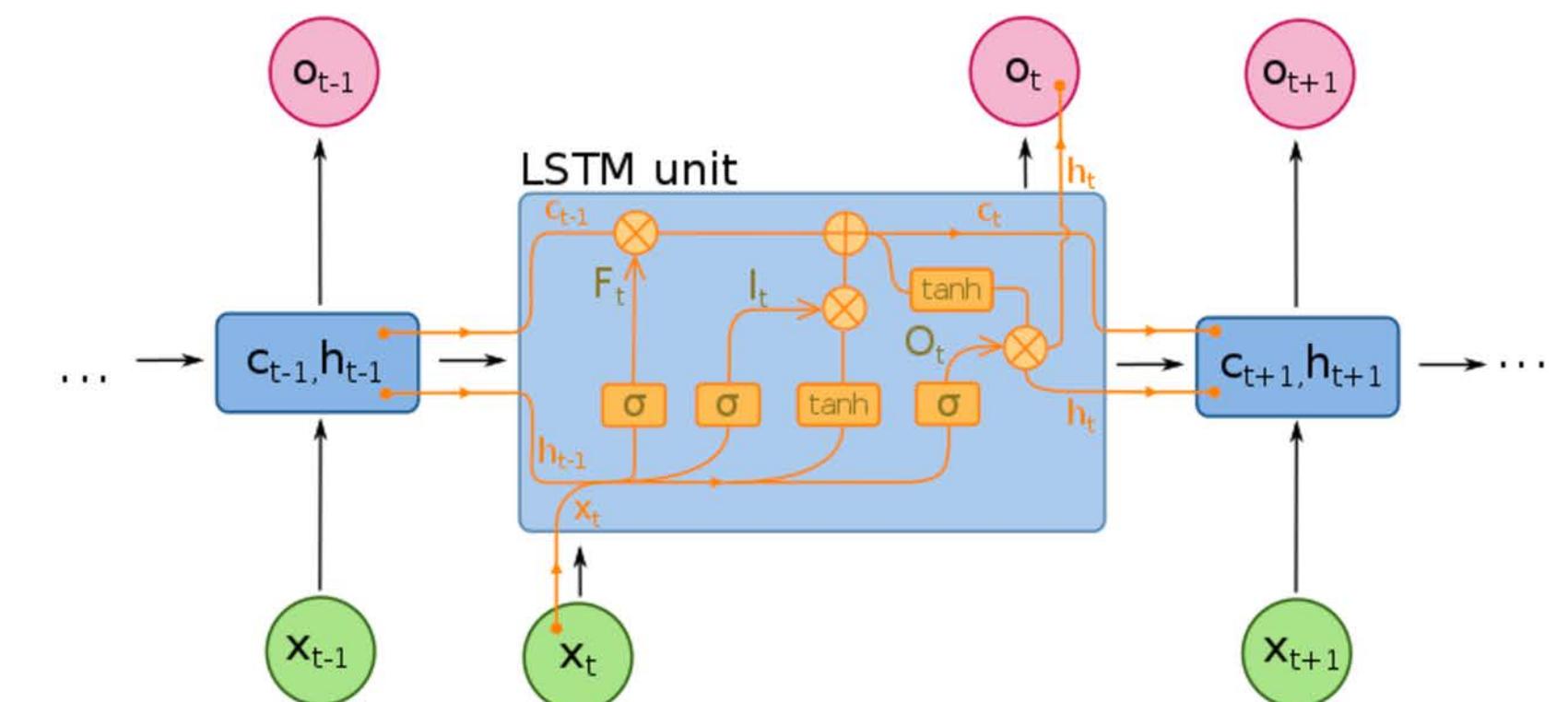
VPL + AIプログラミング

VPLで手軽にプログラミング



AIのローエンドの動きを可視化できる

- 分かりにくい値の動き
- 重みやフィルタの働き
- データの流れ



https://ja.m.wikipedia.org/wiki/%E3%83%95%E3%82%A1%E3%82%A4%E3%83%AB:Long_Short-Term_Memory.svg

脆弱性を発見、対応できる

- 値の動きや処理の流れがわかる
 - → 脆弱性の発見、対応の検討がつく

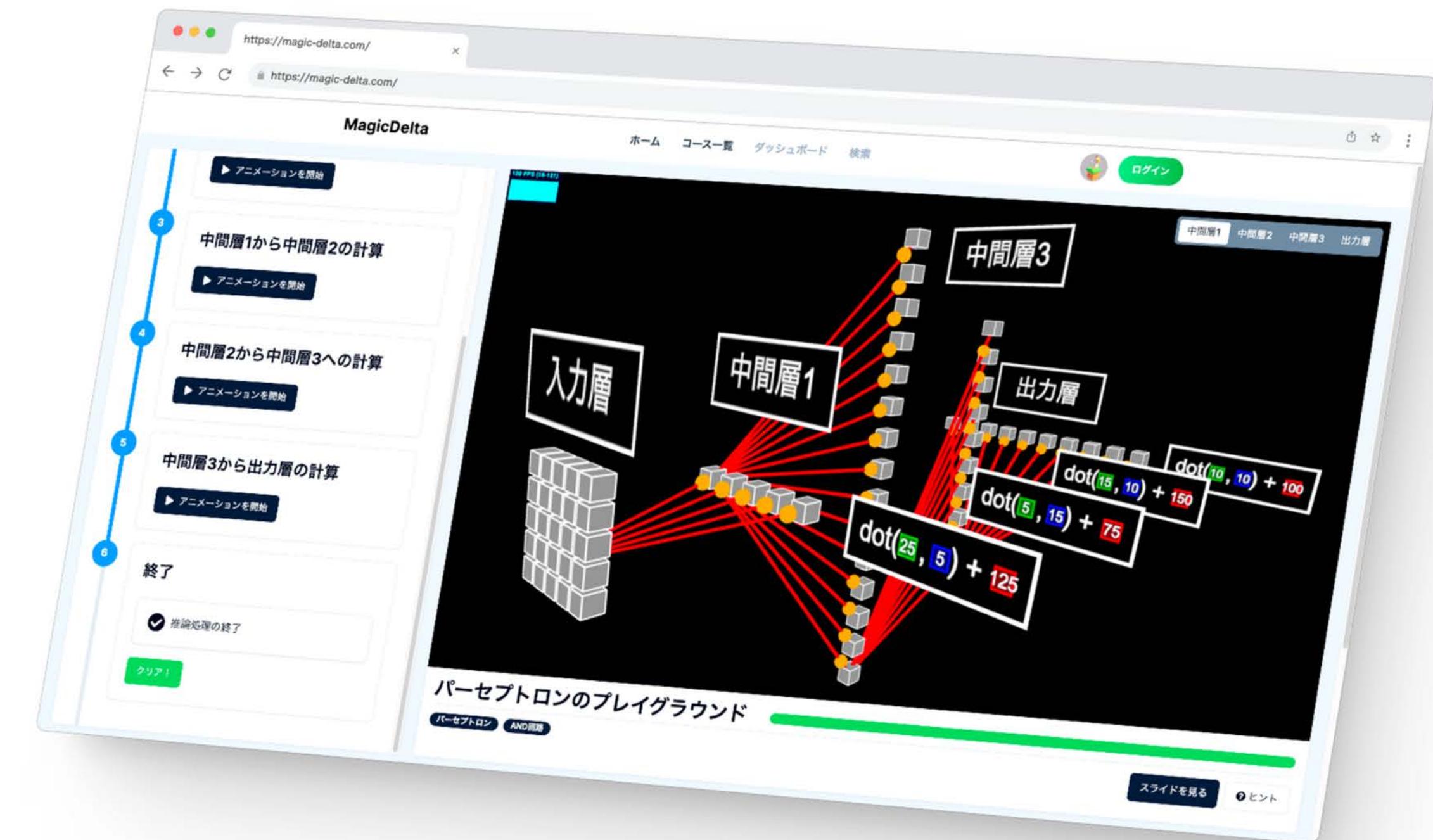
脆弱性について学べるサービスに

- AIの基礎を勉強できる
- セキュリティについて学べる
 - セキュアなAIの開発
 - AIの脆弱性に対応できるプログラマーを増やす

まとめ

- 学習サービスを個人開発
- 自作アニメーションエンジンの開発
 - AIの脆弱性を可視化

ご清聴ありがとうございました



秋穂 正斗