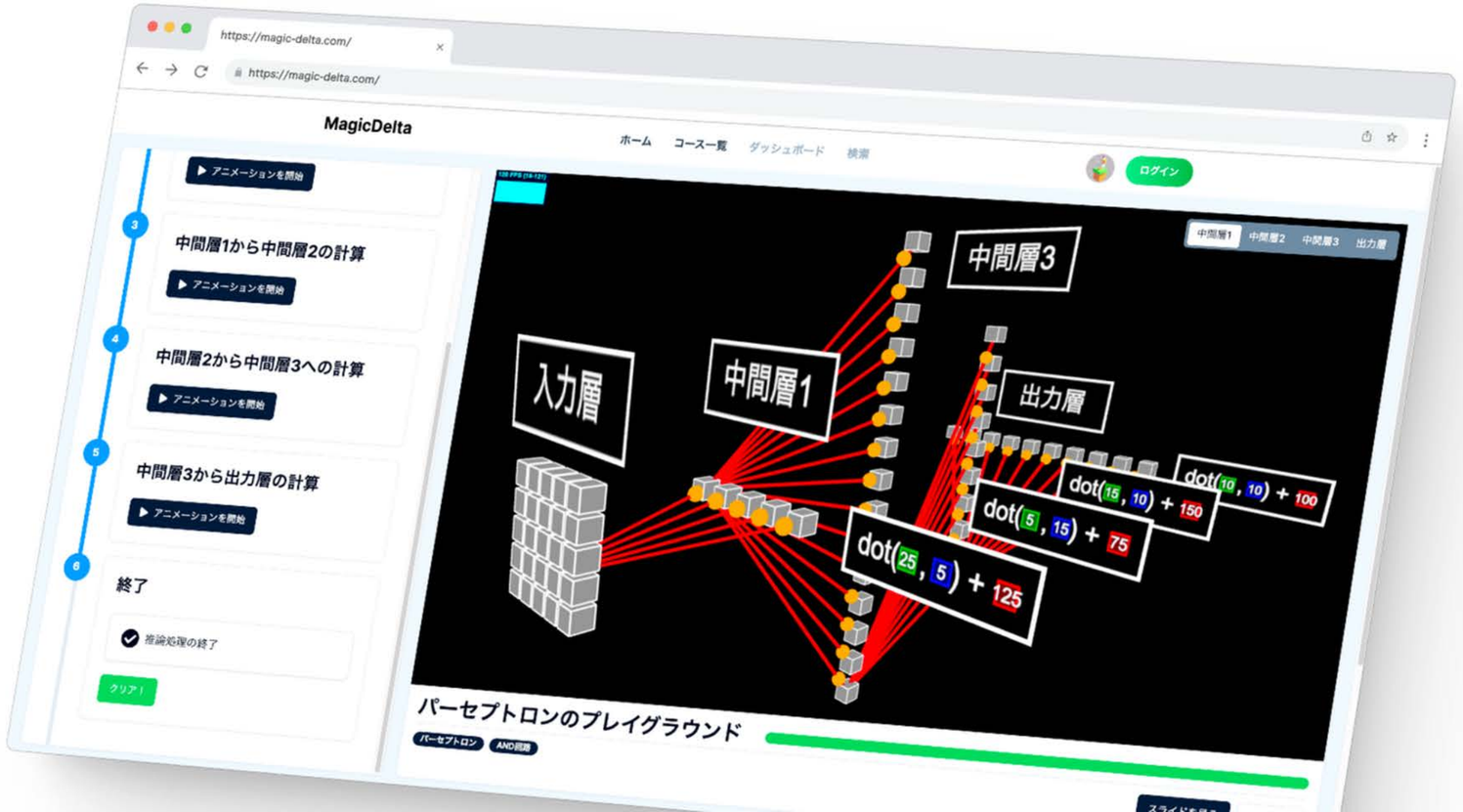
2024 SecHack365 開発駆動コース 仲山ゼミ

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

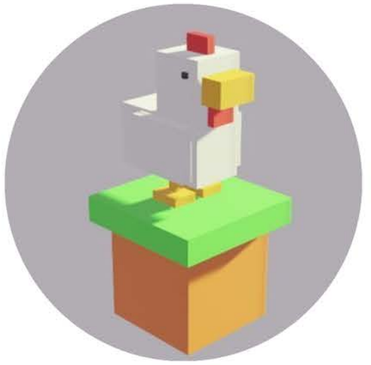
秋穂 正斗（手羽先）

- 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

バックグラウンド



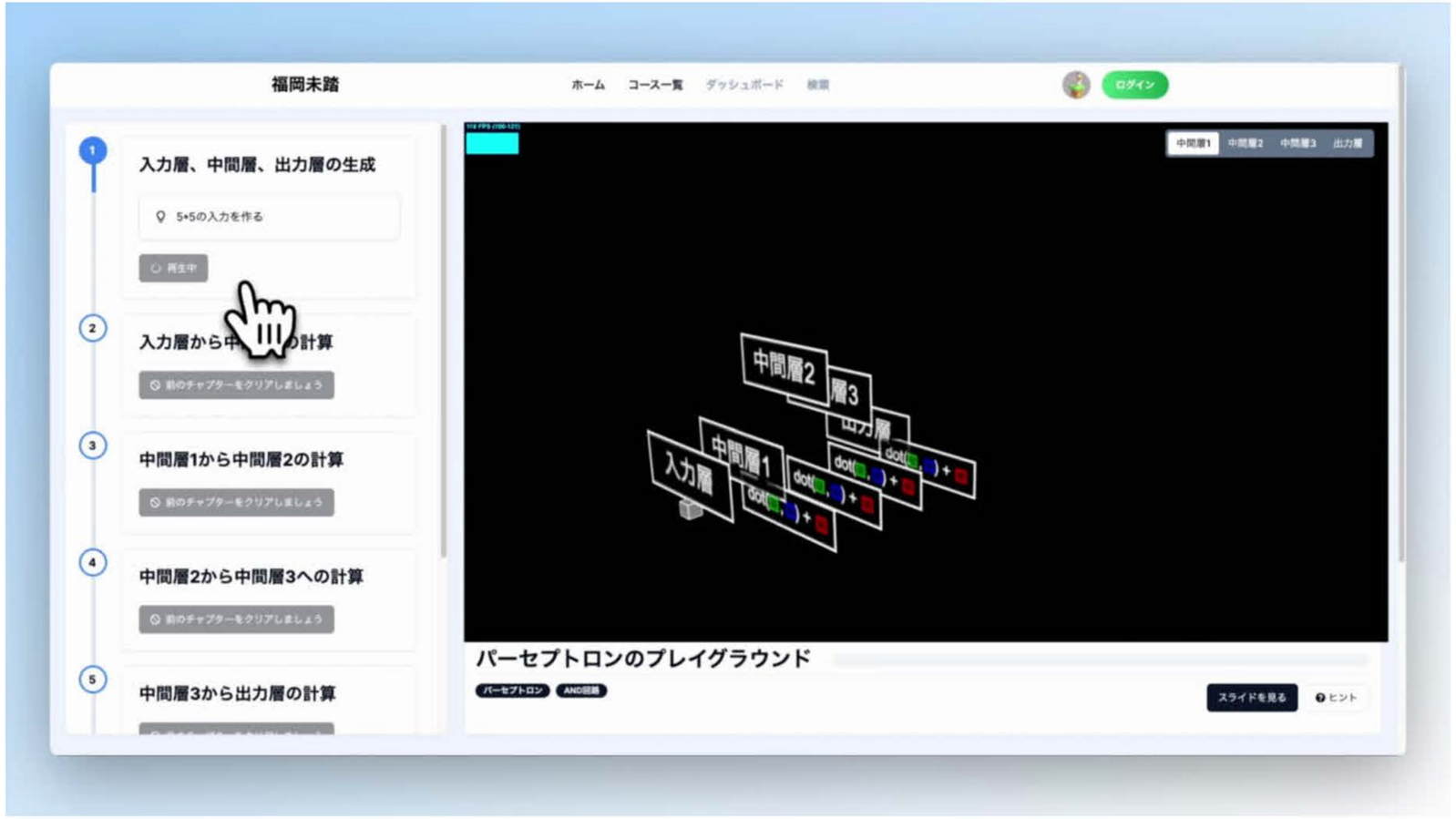
- **小学5年**
  - Scratchを始める
- **高校1年**
  - プログラミング
- **高校2年**
  - AI
  - 大学数学を独学
- **今年3月にN高を卒業**

高校2年生をAIに溶かした

開発駆動コース

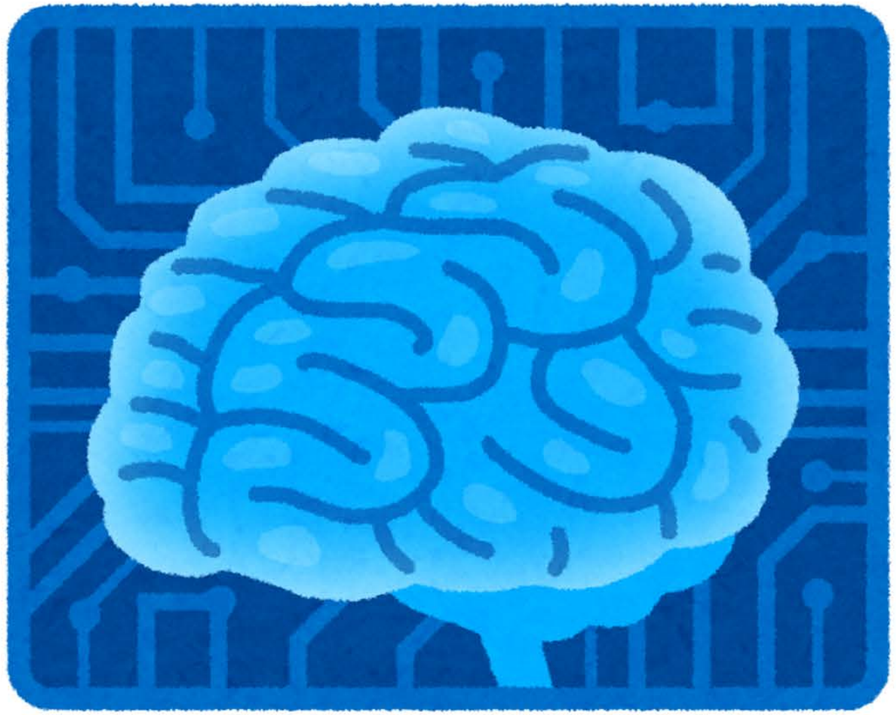
- **まず開発をしてみて、作りたいものを決める**
  - 開発が原動→開発駆動
- **実装の割合が多い**
  - 考える、まとめるよりもゴリゴリ開発するイメージ

作りたいもの



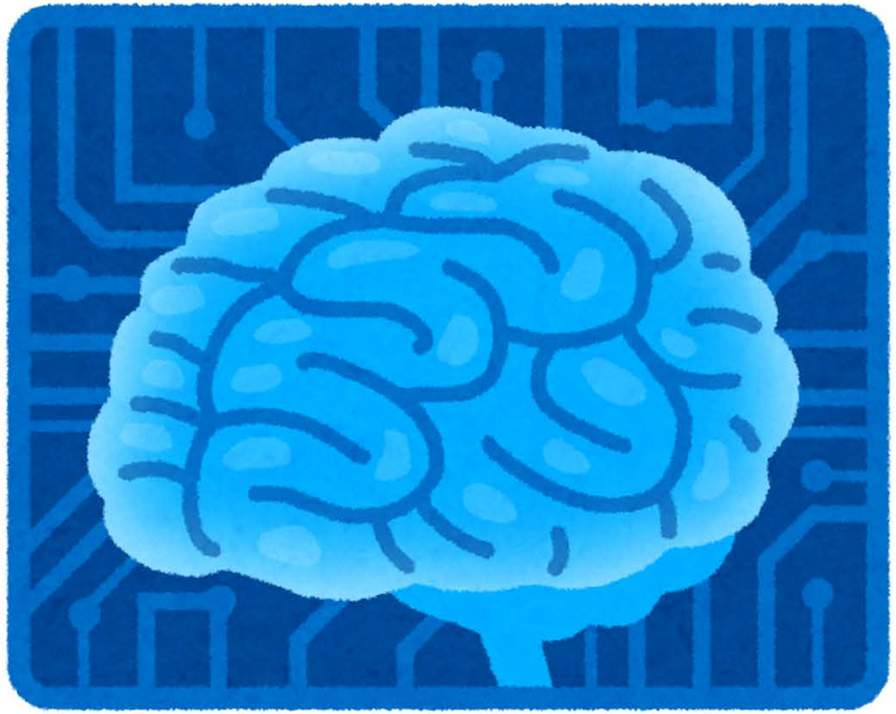
- AIの脆弱性を理解できるWebサービス
- ゲーム感覚で学習 / 実験できる

画像認識をするAI



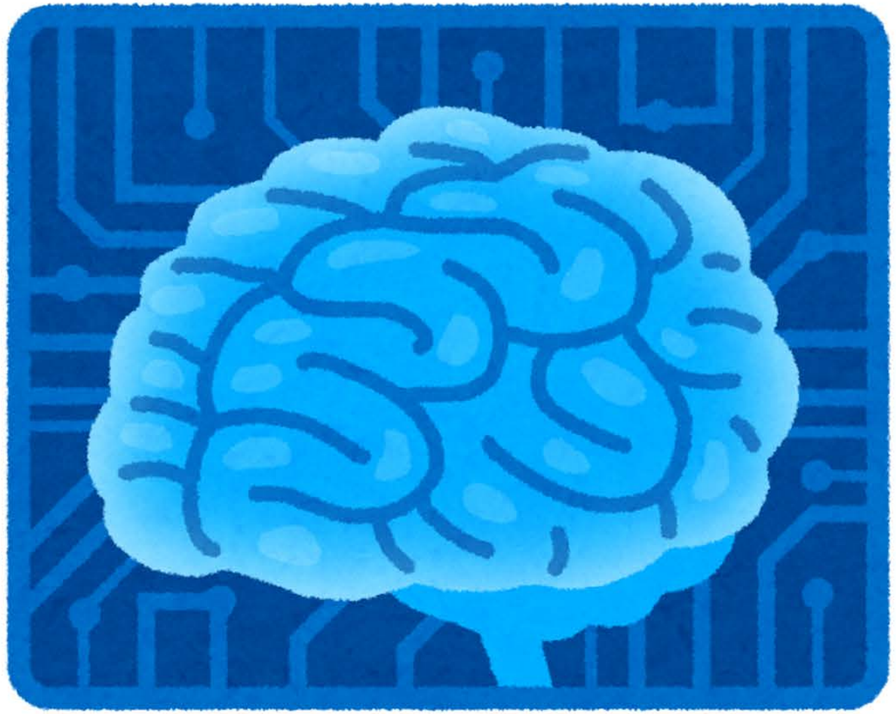
[2] Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy. “Explaining and Harnessing Adversarial Examples.” arXiv:1412.6572

テナガザル！

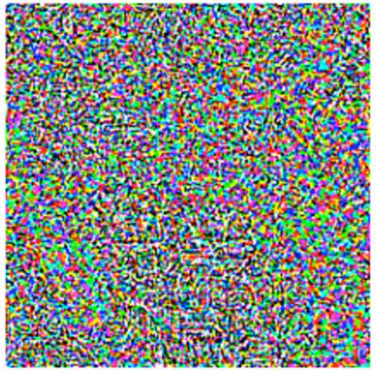


[2] Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy. “Explaining and Harnessing Adversarial Examples.” arXiv:1412.6572

パンダ！

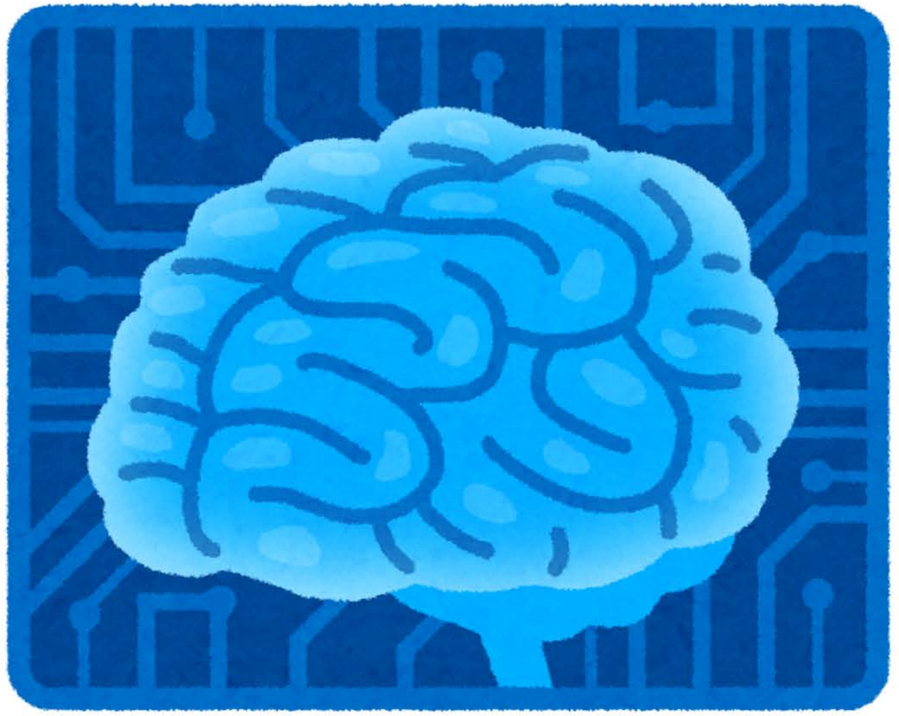


[2] Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy. “Explaining and Harnessing Adversarial Examples.” arXiv:1412.6572



+

テナガザル！



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

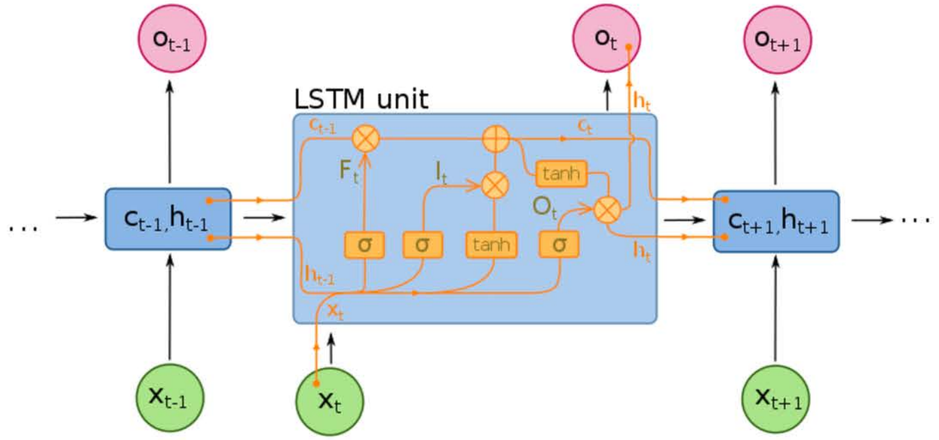
AIの脆弱性の危険性

- **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

[1707.08945] Robust Physical-World Attacks on Machine Learning Models

AIの仕組みは難しい

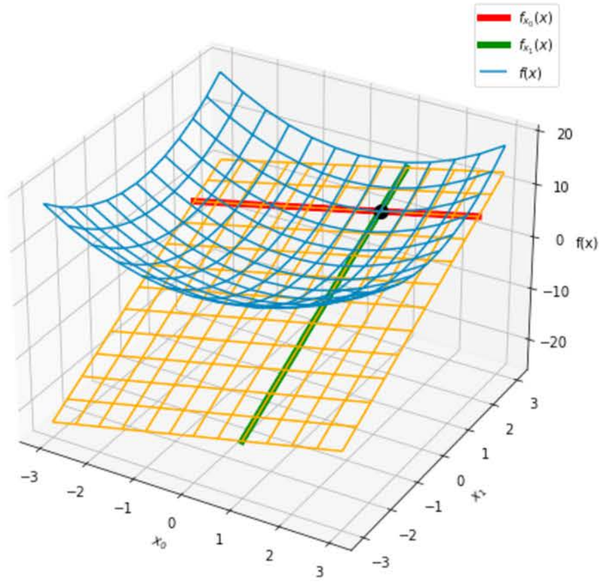
- ビジョンや理念
- ターゲット



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

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

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




```python
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
```

$$\frac{\partial L}{\partial w_{1,1}} = \frac{\partial L}{\partial y_{1,1}}\frac{\partial y_{1,1}}{\partial w_{1,1}} + \frac{\partial L}{\partial y_{2,1}}\frac{\partial y_{2,1}}{\partial w_{1,1}} + \cdots + \frac{\partial L}{\partial y_{N,1}}\frac{\partial y_{N,1}}{\partial w_{1,1}}$$

$$= \sum_{h=1}^{H} \frac{\partial L}{\partial y_{n,1}}\frac{\partial y_{n,1}}{\partial w_{1,1}}$$

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

書籍や動画で出来ないことは？

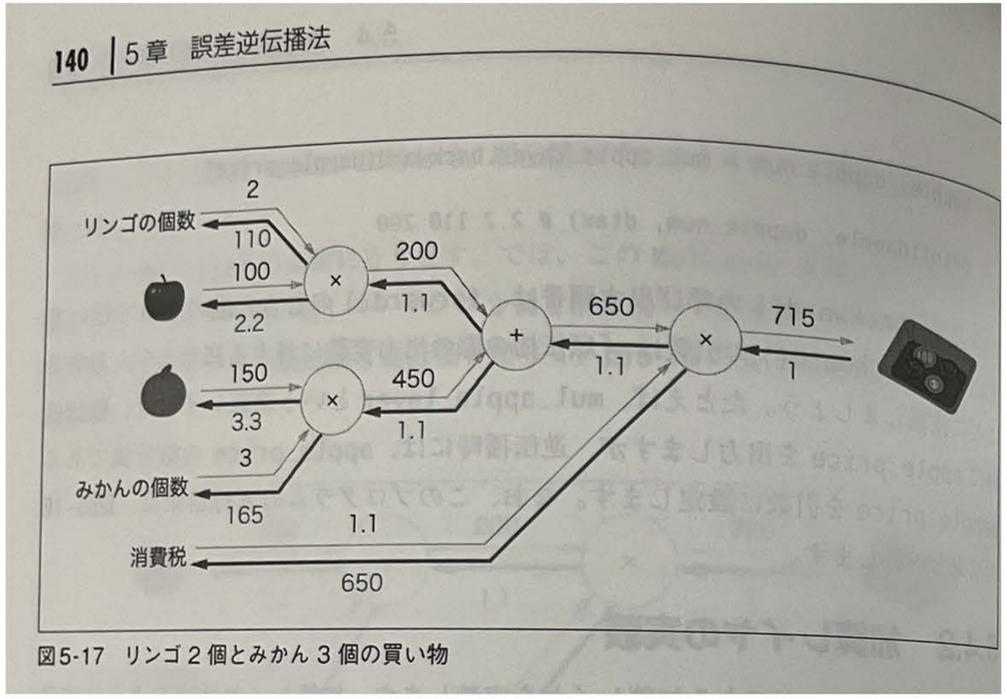
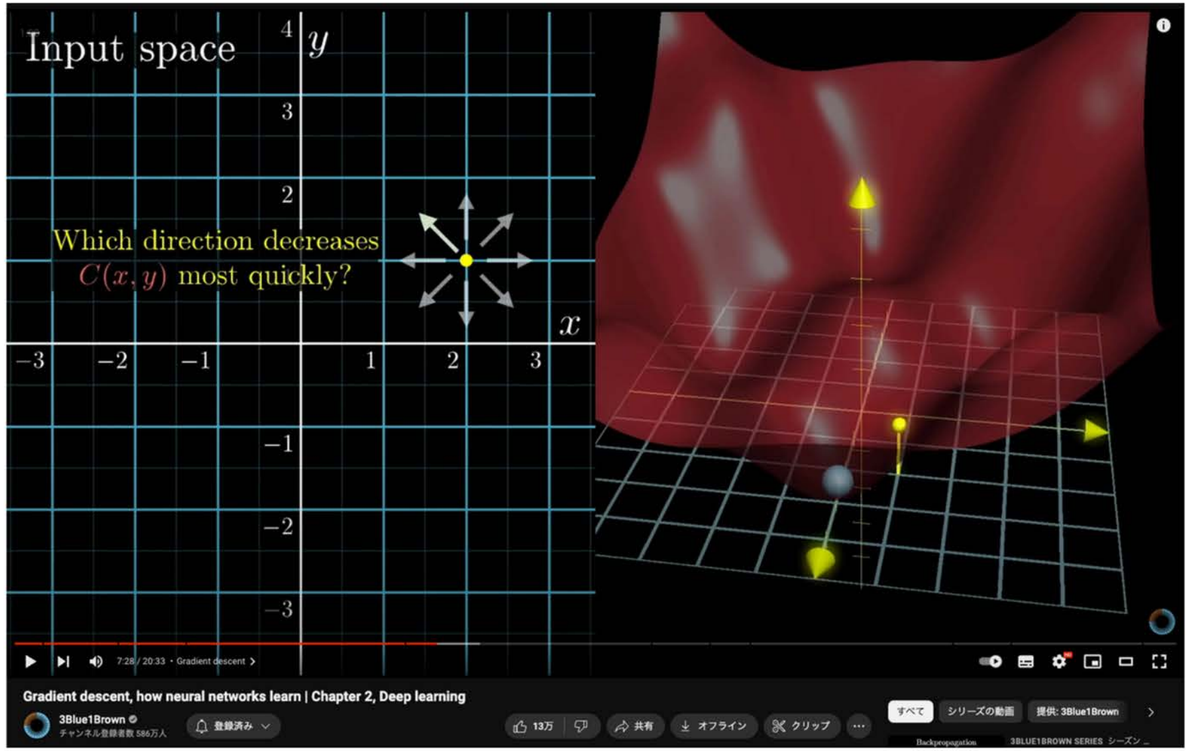


図5-17 リンゴ２個とみかん３個の買い物

ゼロから作るDeep Learning



https://www.youtube.com/watch?v=IHZwWFHWa-w

→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

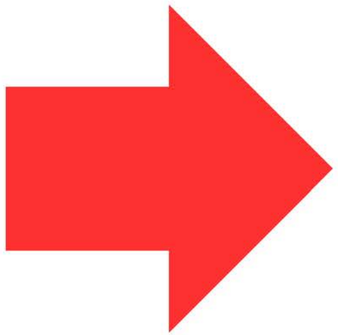
アニメーションで可視化する

視点や位置を自由に動かせる

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: 0xffff

// 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(neuronRadi
      const neuronMaterial = new THREE.MeshBasicMaterial({ color
      const neuron = new THREE.Mesh(neuronGeometry, neuronMateri
      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; layerIn
  for (let rowIndex = 0; rowIndex < layers[layerIndex].length; r
    for (let neuronIndex = 0; neuronIndex < layers[layerIndex][r
```

約100~500行のコード

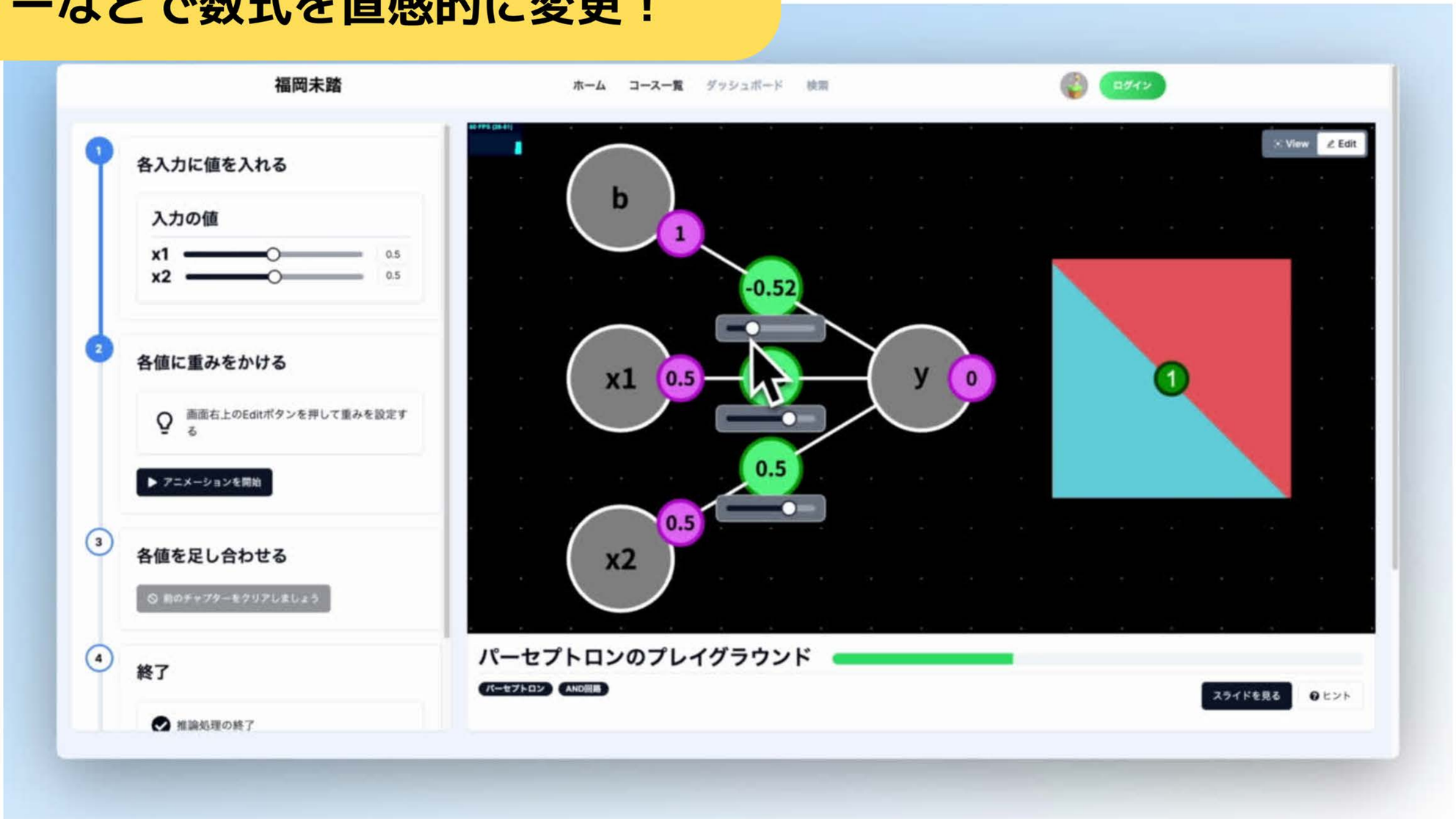


```
const layer = [
      { y: 5, x: 5, gap: 0.5},
      { y: 1, x: 5 },
      { y: 15, x: 1 },
      { y: 10, x: 1 },
      { y: 1, x: 10 },
];

create3DLayer(layer);
```

実際の10行のコード

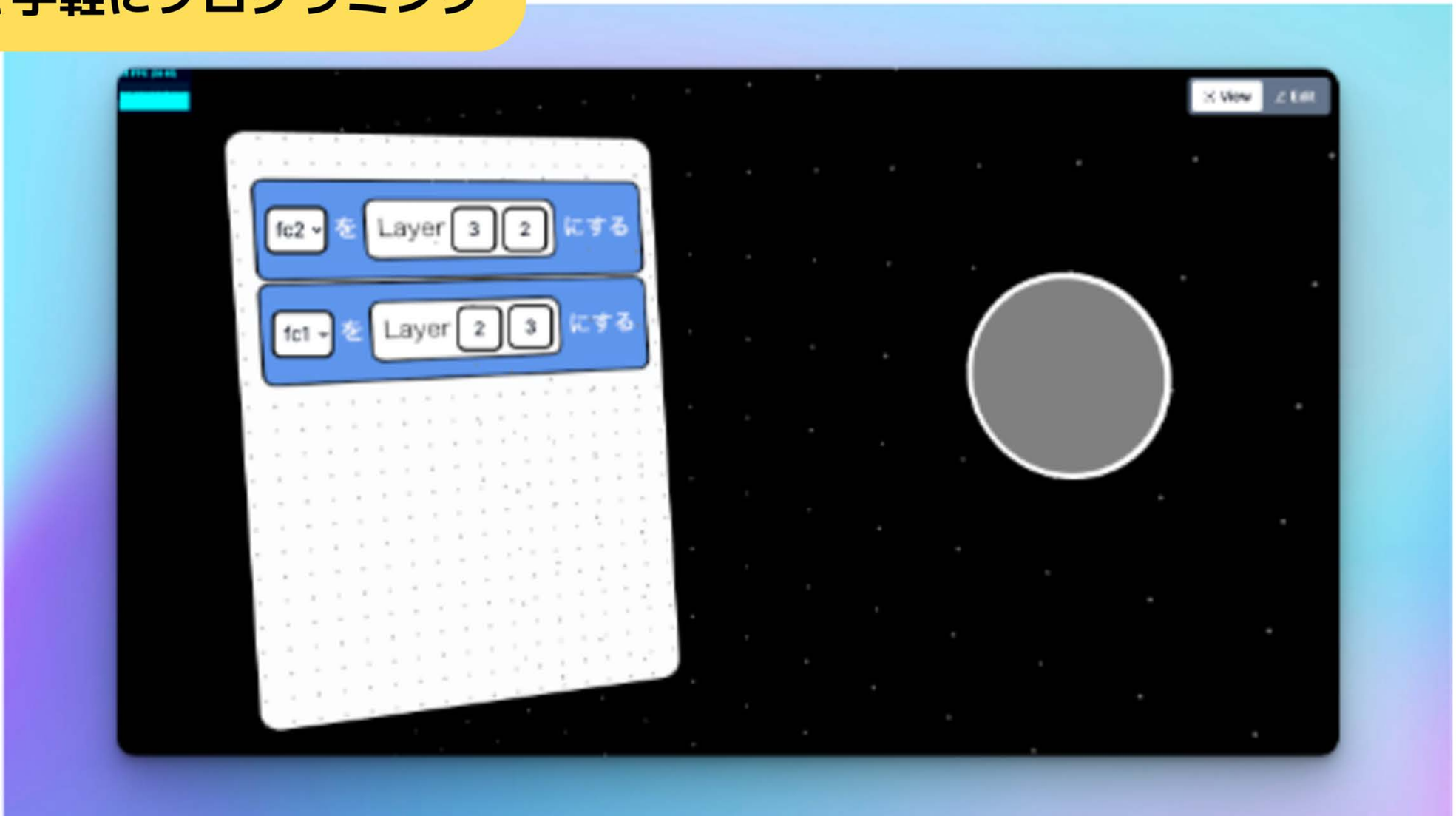
直感的に理解できる

スライダーなどで数式を直感的に変更！



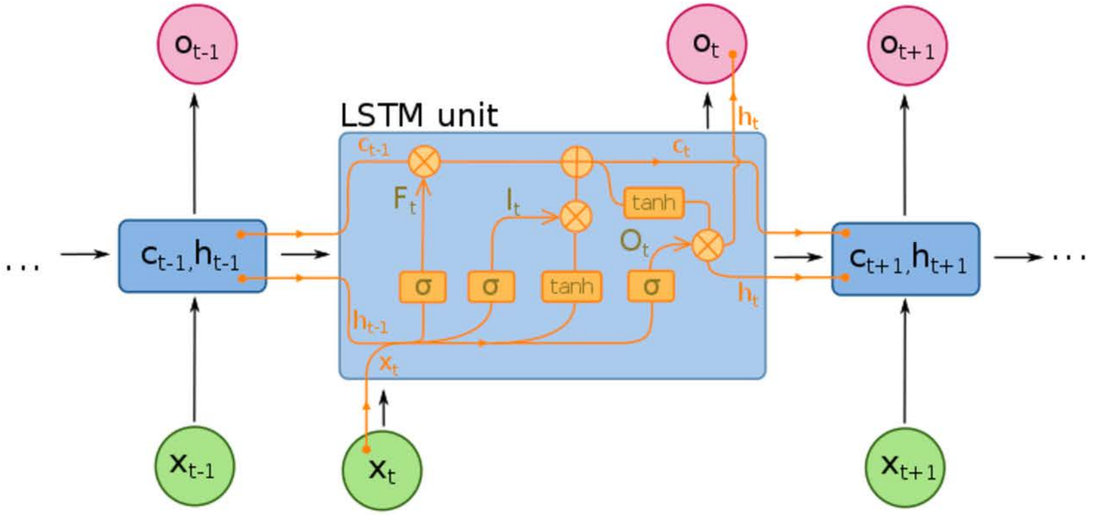
VPL + AIプログラミング

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

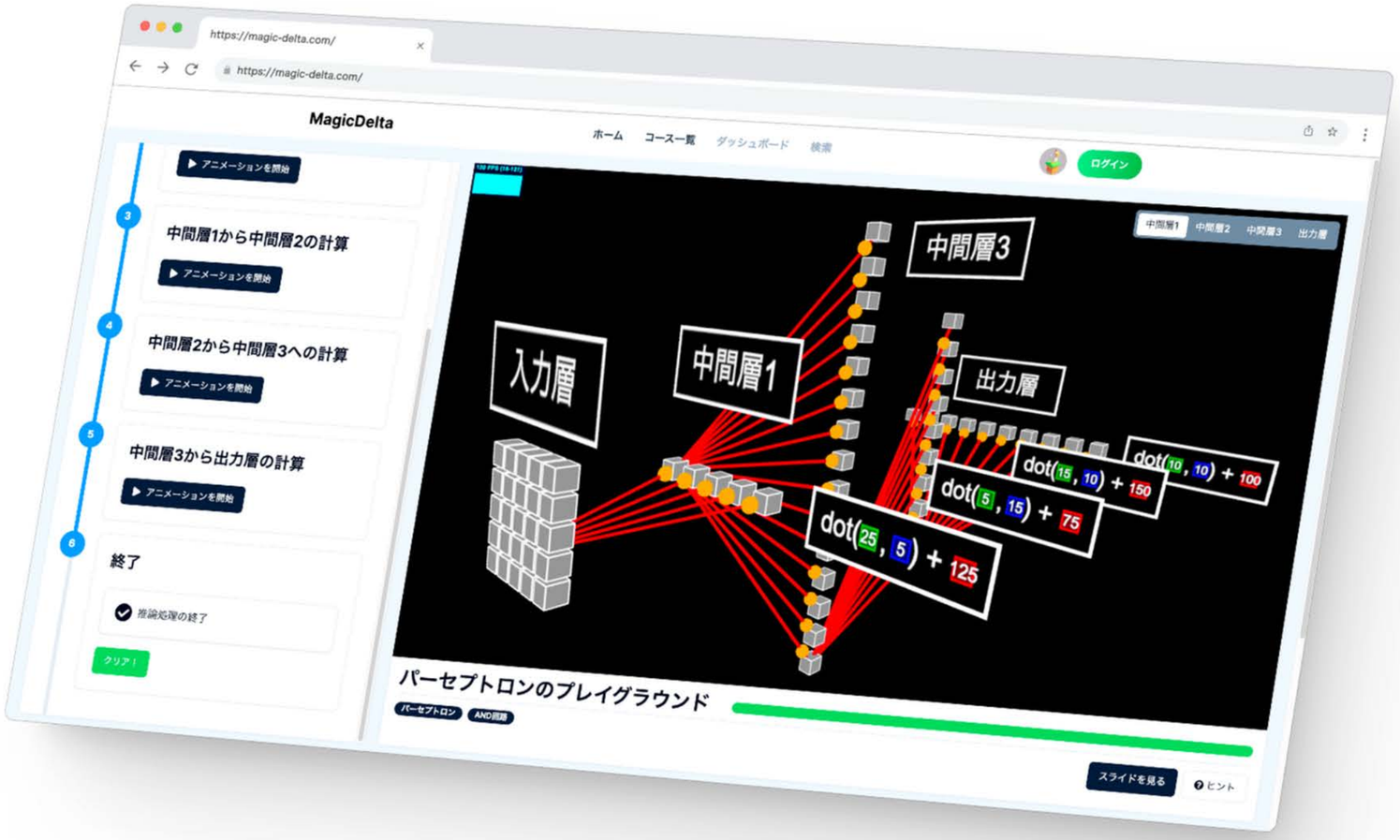
脆弱性について学べるサービスに

- **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