メディアアート・プログラミング
ml5.js 実践 – 転移学習3：転移学習を活用してゲームを作る

2019年11月6日

前橋工科大学総合デザイン工学科
田所 淳

今日の内容

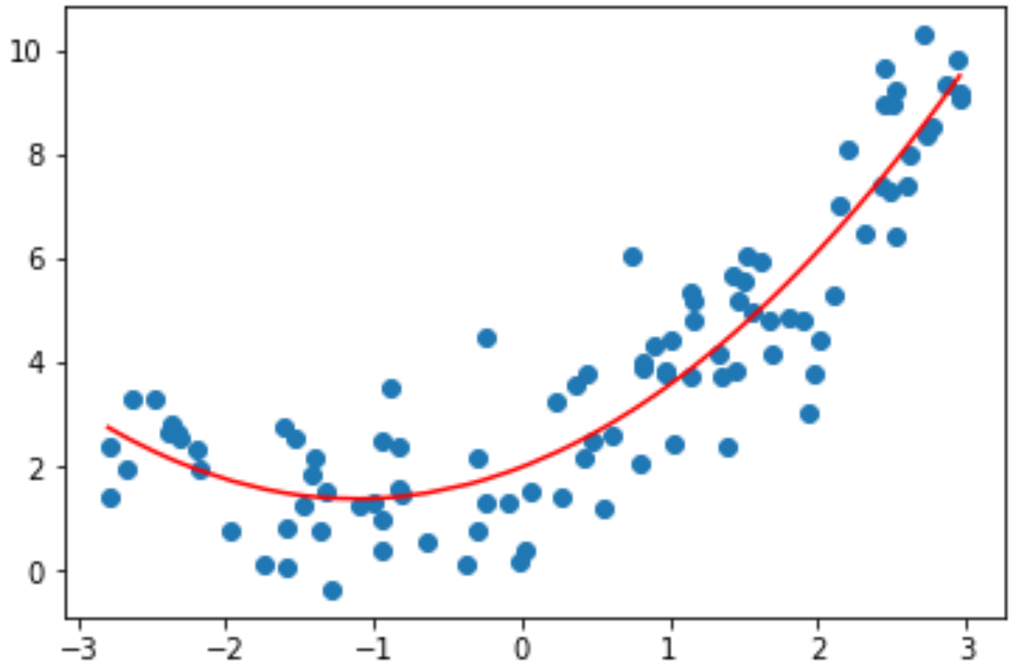
- 転移学習を活用してゲームを作る
  - 先週の画像の特徴量抽出による画像の回帰分析のプログラムを元に…
  - ジェスチャーを使ってコントロールするゲームをつくる!

- (もし時間があれば)
  - 作成したml5.js + p5.jsのプログラムを公開するための準備
  - OpenProcessingに登録
  - プログラムの公開を試してみる

先週の復習: 画像の特徴量抽出による回帰分析

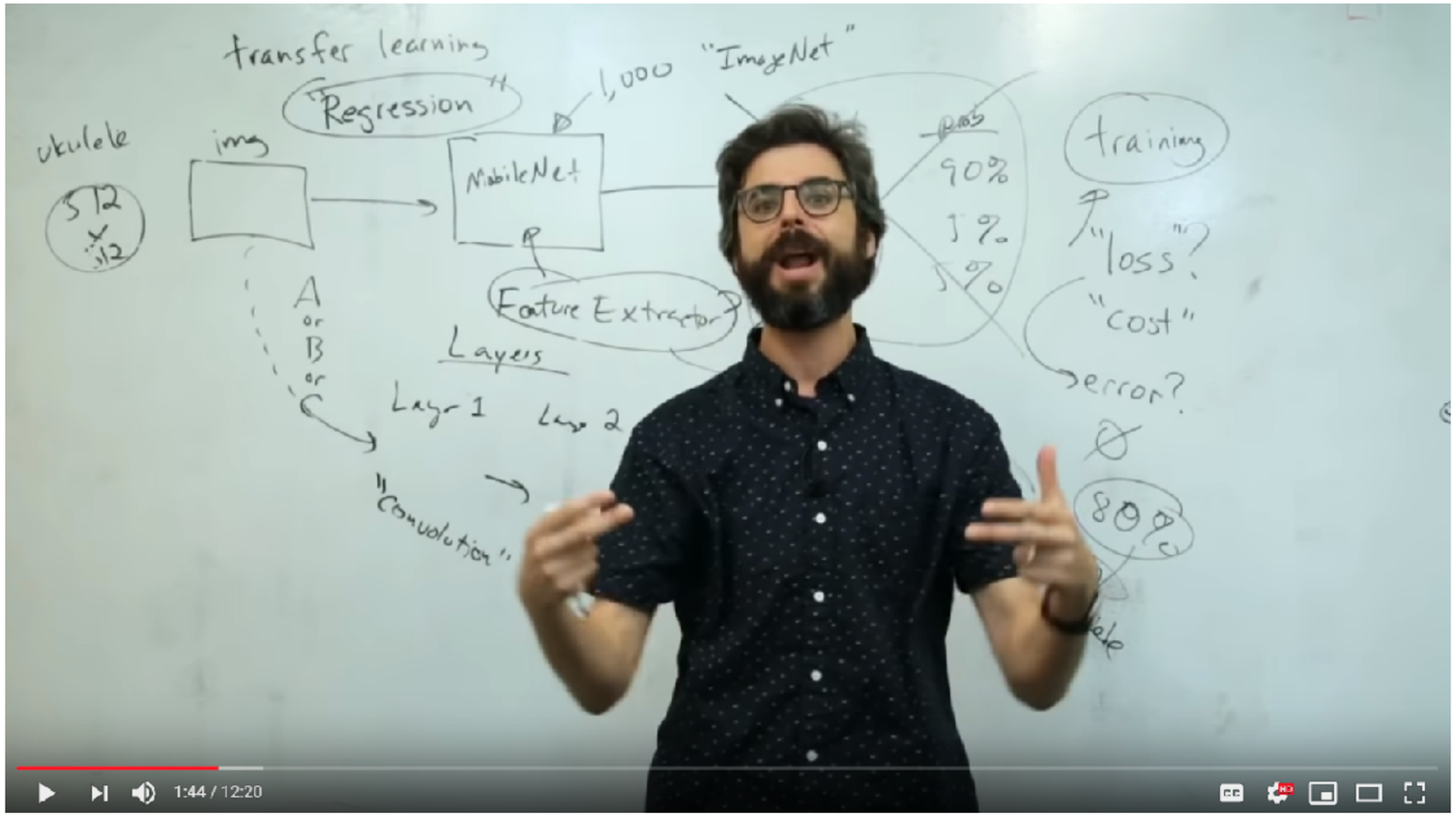
回帰分析 (Regression) とは?

- 回帰分析 (Regression) とは何か?
- 回帰分析：
  - 相関関係や因果関係があると思われる2つの変数のうち、一方の変数から将来的な値を予測するための予測式（回帰直線）を求めるための手法。



回帰分析 (Regression) とは?

- 前回の特徴量抽出によるクラス分けと回帰分析はどう違うのか
- 今回もDan Shiffman先生の解説を参考にしてみる
- https://youtu.be/aKgq0m1YjvQ



m5js 転移学習：特徴量抽出による画像の回帰分析

m5js 転移学習：特徴量抽出による画像の回帰分析

- 今回は、前回の「特徴量抽出による画像のクラス分け」のサンプルを改造します
- 以下からダウンロード (or コピペ)
- https://gist.github.com/tado/a27bf33ece7c96e1101c3ff330337c1e

sketch.js　　Raw

```
/*
 *  メディアアートプログラミング
 *  転移学習の特徴抽出による画像のクラス分けサンプル
 *  2019.10.23
 */

let featureExtractor;
let classifier;
let video;
let loss;
let dogImages = 0;
let catImages = 0;
let badgerImages = 0;
let status = ''; //現在の状態を左上に表示
let result = ''; //クラス分け結果を中央に表示

function setup() {
  //キャンバスを画面全体に
  createCanvas(windowWidth, windowHeight);
  video = createCapture(VIDEO);
  video.size(320, 240);
```

m5js 転移学習：特徴量抽出による画像の回帰分析

- 改造するポイント
  - キーによる操作ではなく、GUIのスライダーとボタンによる操作へ
  - クラス分けから回帰分析へ
    - featureExtractor.classification → featureExtractor.regression
  - スライダーの値と画像を組にして追加していく
  - ラベル名を表示するのではなく、回帰分析のスコアを表示するように

m5js 転移学習：特徴量抽出による画像の回帰分析

- sketch.js

```
let featureExtractor;
let regressor; //classifierをregressorに
let video;
let loss;
let status = ''; //現在の状態を左上に表示
let showResult = ''; //クラス分け結果を中央に表示
let slider; //スライダー
//ボタン
let addImageButton, trainButton, predictButton;

function setup() {
  createCanvas(windowWidth, windowHeight);
  video = createCapture(VIDEO);
  video.size(320, 240);
  video.hide();
  //特徴量抽出
  featureExtractor
  = ml5.featureExtractor('MobileNet', modelReady);
  const options = { numLabels: 3 };
  //classificationからrecressionへ
  regressor = featureExtractor.regression(video);
  //スライダーを配置
  slider = createSlider(0.0, 1.0, 0.5, 0.01);
  slider.position(20, 40);

//ボタンを配置
  addImageButton = createButton('add image');
  addImageButton.position(slider.x+slider.width+20,40);
  addImageButton.mousePressed(addImage);
  trainButton = createButton('train');
  trainButton.position(addImageButton.x+addImageButton.
width+5,40);
  trainButton.mousePressed(train);
  predictButton = createButton('start predict');
  predictButton.position(trainButton.x+trainButton.widt
h + 5,40);
  predictButton.mousePressed(predict);
}
```

m5js 転移学習：特徴量抽出による画像の回帰分析

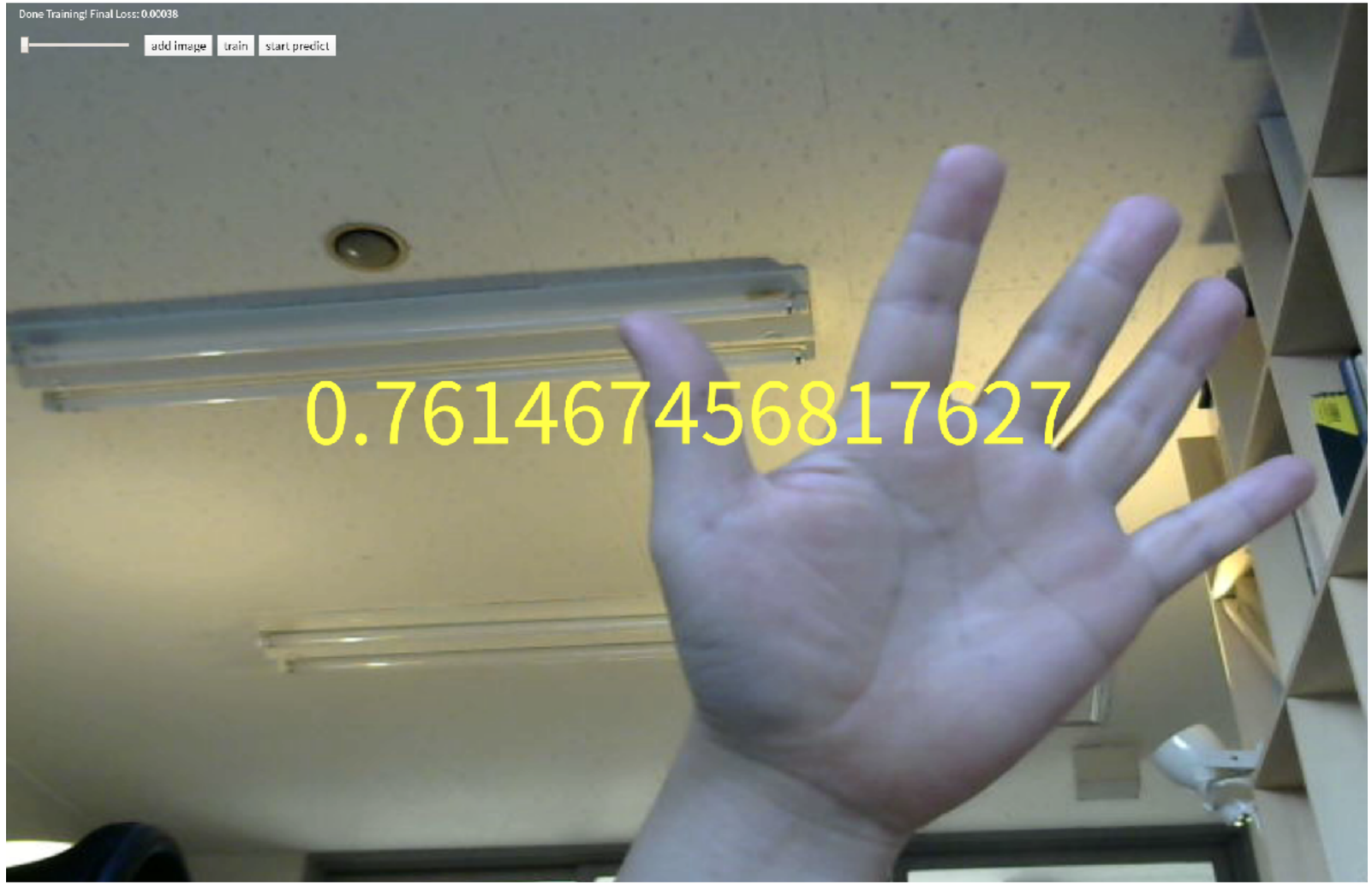
- sketch.js

```
function draw(){
  background(0);
  //ビデオ映像をフルクスリーン表示
  image(video, 0, 0, width, height);
  //現在の状態(status)を表示
  fill(255);
  textSize(12);
  textAlign(LEFT);
  text(status, 20, 20);
  //回帰分析の結果を表示
  fill(255, 255, 0);
  textSize(100);
  textAlign(CENTER);
  text(showResult, width/2, height/2);
}

//画像を追加
function addImage(){
  regressor.addImage(slider.value());
}
//訓練開始
function train(){
  regressor.train(function(lossValue) {
    if (lossValue) {
      loss = lossValue;
      status = 'Loss: ' + loss;
    } else {
      status = 'Done Training! Final Loss: ' + loss;
    }
  });
}
```

m5js 転移学習：特徴量抽出による画像の回帰分析

- 完成!!



画像の特徴量抽出による回帰分析をゲームに活用!

画像の特徴量抽出による回帰分析をゲームに活用!

- 画像の特徴量抽出による回帰分析をゲームに活用してみる

- まずは元になるゲームを作る
  - 方針: メインのml5.jsによる画像の回帰分析と機能を分割する
  - → できるだけクラスによるモジュール化を行う

- 今回はシンプルな例として、テニスゲームを作成しました!
  - まずはテニスゲームの部分のみ作成

画像の特徴量抽出による回帰分析をゲームに活用!

- TennisGameクラス
  - ゲーム全体の統合
  - ボールとバー(ラケット)を配置
  - ボールがバーに当ったら跳ね返るように

- Ballクラス
  - ボールの動きと表示
  - 壁にぶつかったらバウンドする

- Barクラス
  - ボールを打ち返すバー
  - 左右に移動

画像の特徴量抽出による回帰分析をゲームに活用!

- 制作テンプレート
- 制作のためのテンプレートを用意しました、以下からダウンロード

http://bit.ly/2NOv98q

画像の特徴量抽出による回帰分析をゲームに活用!

- テニスゲーム:

```
class TennisGame {
  constructor(){
    this.ball = new Ball();
    this.bar = new Bar();
    this.score = 0;
  }
  update(){
    this.ball.update();
    // バー(コントローラー)でバウンド
    if (this.ball.location.y > height - 50) {
      if (abs(this.bar.location.x - this.ball.location.x) < 150) {
        this.ball.velocity.y *= -1.0;
        this.ball.velocity.mult(1.2);
        this.score += 100;
      } else {
        this.initGame();
      }
    }
  }
  draw(){
    fill(255);
    this.ball.draw();
    this.bar.draw();
    textSize(40);
    text(this.score, 20, 60);
  }
  initGame() {
    // リセット
    this.ball.location.set(width / 2, 40);
    this.ball.velocity = createVector(random(-2, 2), 2);
    //this.barX = width / 2;
    this.bar.location.x = width / 2;
  }
}

class Ball {
  constructor() {
    this.location = createVector(width / 2, 40);
    this.velocity = createVector(random(-2, 2), 2);
  }
  update() {
    this.location.add(this.velocity);
    // 壁でバウンド
    if (this.location.x < 0 || this.location.x > width) {
      this.velocity.x *= -1;
    }
    if (this.location.y < 0) {
      this.velocity.y *= -1;
    }
  }
  draw() {
    fill(255);
    noStroke();
    ellipse(this.location.x, this.location.y, 10, 10);
  }
}

class Bar {
  constructor() {
    this.location = createVector(width / 2, height - 50);
    this.speed = 10;
  }
  draw() {
    fill(255);
    noStroke();
    rectMode(CENTER);
    rect(this.location.x, this.location.y, 300, 20);
  }
  moveLeft(){
    this.location.x -= this.speed;
  }
  moveRight(){
    this.location.x += this.speed;
  }
}
```

画像の特徴量抽出による回帰分析をゲームに活用!

- TennisGameをp5.jsのメインのループ (setupとdraw) に読み込み
- シンプルなテニスゲームを完成させる

画像の特徴量抽出による回帰分析をゲームに活用!

- テニスゲーム:

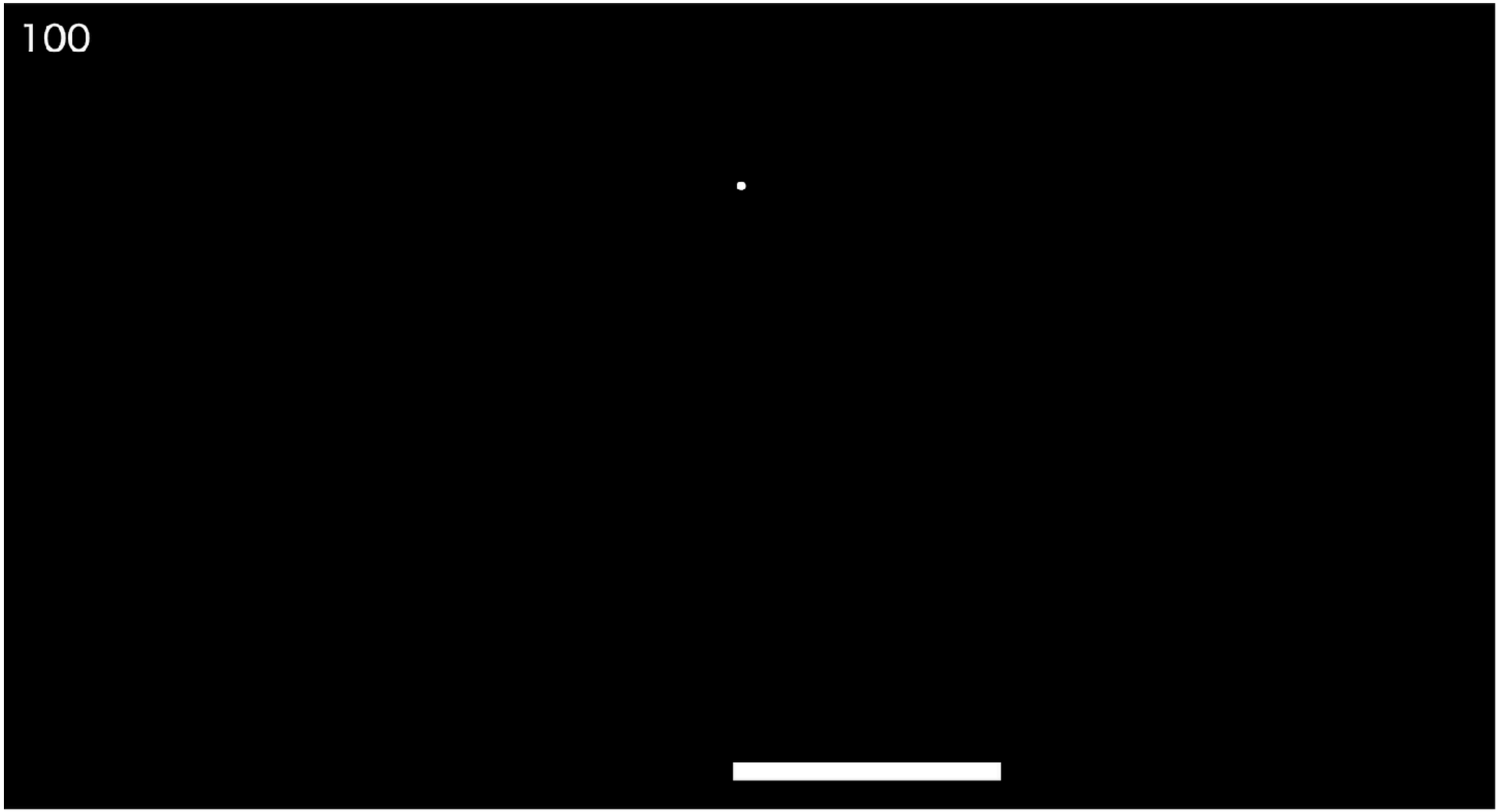
```
let tennis;

function setup(){
  createCanvas(windowWidth, windowHeight);
  frameRate(60)
  //ゲーム初期化
  tennis = new TennisGame();
}

function draw() {
  background(0);
  //ゲームの更新と表示
  tennis.update();
  tennis.draw();
  //キー操作でbarを左右に
  if(keyIsPressed && keyCode === LEFT_ARROW){
    tennis.bar.location.x -= 10;
  }
  if(keyIsPressed && keyCode === RIGHT_ARROW){
    tennis.bar.location.x += 10;
  }
}
```

画像の特徴量抽出による回帰分析をゲームに活用!

- 完成!!



画像の特徴量抽出による回帰分析をゲームに活用!

- いよいよml5の画像回帰分析のプログラムと統合!
- 分析結果 (showResult) をbarの左右の移動に対応される
- 学習中はゲームを開始しない工夫
  - 変数modeを用意してモードを切り替える
  - 0：学習中
  - 1：ゲーム開始&プレイ
- TennisGame、Ball、Barのそれぞれのクラスはそのまま活用する

画像の特徴量抽出による回帰分析をゲームに活用!

- テニスゲーム:

```
let featureExtractor, regressor, loss, predictResult=0.5;
let video;
let status = 'loading...';
let slider, addImageButton, trainButton, predictButton;
let mode = 0; //ゲームモード 0:学習中、1:ゲームプレイ
let game; //ゲーム本体

function setup() {
  createCanvas(windowWidth, windowHeight);

  //新規にゲームを生成
  game = new TennisGame();

  //解析関係の設定
  video = createCapture(VIDEO);
  video.size(320, 240);
  video.hide();
  featureExtractor
  = ml5.featureExtractor('MobileNet', modelReady);
  regressor = featureExtractor.regression(video);
  slider = createSlider(0.0, 1.0, 0.5, 0.01);
  slider.position(20, 40);
  addImageButton = createButton('add image');
  addImageButton.position(
    slider.x + slider.width + 20, 40);
  addImageButton.mousePressed(addImage);
  trainButton = createButton('train');
  trainButton.position(
    addImageButton.x + addImageButton.width + 5, 40);
  trainButton.mousePressed(train);
  predictButton = createButton('start predict');
  predictButton.position(
    trainButton.x + trainButton.width + 5, 40);
  predictButton.mousePressed(predict);
}

function draw(){
  background(0);
  image(video, 0, 0, width, height);
  fill(0, 63);
  noStroke();
  rectMode(CORNER);
  rect(0, 0, 400, 80);
  fill(255);
  textSize(12);
  textAlign(LEFT);
  text(status, 20, 20);
  //もしゲームモードだったら
  if(mode == 1){
    //ゲームの更新と描画
    game.update();
    game.draw();
    //分析の結果でバーを動かす
    let speed = map(predictResult, 0.0, 1.0, -10, 10);
    game.bar.location.x += speed;
  }
}
```

画像の特徴量抽出による回帰分析をゲームに活用!

- テニスゲーム:

```
//画像を追加
function addImage(){
  regressor.addImage(slider.value());
}

//訓練開始
function train(){
  regressor.train(function(lossValue) {
    if (lossValue) {
      loss = lossValue;
      status = 'Loss: ' + loss;
    } else {
      status = 'Done Training! Final Loss: ' + loss;
    }
  });
}

//回帰分析開始
function predict(){
  //結果が出たらgotResultsを実行
  regressor.predict(gotResults);
  mode = 1;
}

//モデルの読み込み完了
function modelReady() {
  status = 'MobileNet Loaded!';
}
```

```
function gotResults(err, result) {
  //エラー表示
  if (err) {
    console.error(err);
    status = err;
  }
  //クラス分けした結果を表示
  if (result && result.value) {
    showResult = result.value;
    predict();
  }
}
```

画像の特徴量抽出による回帰分析をゲームに活用!

- 完成!!

今日はここまで!