Tensorflowjs/Tensorflow.js

Tensorflowjs

- Python 모듈이며 Tensorflow 모델을 Javascript에서 사용할 수 있는 모델로 변환
- 변환된 모델은 model.json, group1-shard1of1.bin 파일을 포함

Tensorflow.js

- Javascritp 라이브러리리이며 자바스크립트 버전으로 변환된 Tensorflow 모델을 사용하는 라이브러리
- 크롬 확장프로그램에서 실행되는 Javascript에 model.json, group1-shard1of1.bin을 포함시키면 브라우저 기반에서 모델을 사용할 수 있음

크롬 브라우저에서 Hello World 확장프로그램 (Chrome Extension) 작성하기

1. 폴더 및 파일 생성

```
hello-world-extension/
├── manifest.json
├── content.js
└── icon.png (선택사항)
```

1-1. manifest.json

```
{
  "manifest_version": 3,
  "name": "Hello World Extension",
  "version": "1.0.0",
  "description": "My first Chrome extension",

  "content_scripts": [
    {
      "matches": ["<all_urls>"],
      "js": ["content.js"]
    }
  ]
}
```

1-2. contents.js

```
// 페이지 로드 시 "Hello World" 배너 표시
(function() {
    console.log('Hello World Extension loaded!');

    // Hello World 메시지 박스 생성
    const helloBox = document.createElement('div');
    helloBox.textContent = 'Hello World!';
    helloBox.style.cssText = `
        position: fixed;
        top: 20px;
        right: 20px;
        background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
        color: white;
        padding: 20px 30px;
        border-radius: 10px;
        font-size: 24px;
        font-weight: bold;
        font-family: Arial, sans-serif;
        z-index: 999999;
        box-shadow: 0 4px 15px rgba(0, 0, 0, 0.3);
        cursor: pointer;
        animation: slideIn 0.5s ease-out;
    `;

    // 애니메이션 추가
    const style = document.createElement('style');
    style.textContent = `
        @keyframes slideIn {
            from {
                transform: translateX(400px);
                opacity: 0;
            }
            to {
                transform: translateX(0);
                opacity: 1;
            }
        }
    `;
    document.head.appendChild(style);

    // 클릭하면 사라지게
    helloBox.addEventListener('click', () => {
        helloBox.remove();
    });

    // 페이지에 추가
    document.body.appendChild(helloBox);

    // 5초 후 자동으로 사라지게 (선택사항)
    setTimeout(() => {
        if (helloBox.parentNode) {
            helloBox.style.animation = 'slideIn 0.5s ease-out reverse';
```

```
            setTimeout(() => helloBox.remove(), 500);
        }
    }, 5000);
})();
```

2. Chrome에 설치하기

1. Chrome 열기

2. 확장 프로그램 페이지로 이동

- 주소창에 chrome://extensions/ 입력
- 또는 메뉴(3점 아이콘) -> 도구 더보기 -> 확장 프로그램

3. 개발자 모드 활성화

- 우측 상다의 "개발자 모드" 토글 ON

4. 확장 프로그램 로드

- [압축해제된 확장프로그램 로드] 버튼 클릭
- hello-world-extension 폴더 선택

5. 완료

- "확장 프로그램 로드됨" 메시지가 브라우저 하단에 잠시 표시됨
- 확장 프로그램이 목록에 표시됩니다

3. 테스트하기

1. 아무 웹사이트나 방문한다(예: google.com)

2. 페이지 우측 상단에 "Hello World!" 메시지가 나타나는지 확인

3. 메시지를 클릭하면 사라짐

Python기반에서 Tensorflow 모델 생성

- 파이썬 가상환경 생성

```
# 1. 새로운 가상환경 생성 (파이썬 3.11 지정)
conda create -n tfjs_fix python=3.11 -y

# 2. 가상환경 활성화
conda activate tfjs_fix
```

```
# 3. 필수 패키지만 깔끔하게 설치
pip install tensorflow==2.15.0 tensorflowjs==4.17.0
```

- 문제의 원인인 Decision Forests 모듈 제거

```
# 1. 활성화된 가상환경인지 확인 (tfjs_fix)
# 2. 오류를 일으키는 Decision Forests 모듈 제거
pip uninstall tensorflow-decision-forests -y
```

- y = 2x + 1 직선의 방정식에 노이즈를 추가한 데이터 회귀모델 생성

```
import numpy as np
import tensorflow as tf
from tensorflow import keras

# 1. 데이터 생성 (y = 2x + 1)
# -10부터 10까지 100개의 데이터를 생성합니다.
X = np.linspace(-10, 10, 100)
# 실제 수식에 아주 약간의 무작위 노이즈를 더해 학습 효율을 높입니다.
y = 2 * X + 1 + np.random.normal(0, 0.1, X.shape)

# 2. 모델 정의
model = keras.Sequential([
    # 입력 1개, 출력 1개인 가장 단순한 선형 레이어
    keras.layers.Dense(units=1, input_shape=(1,), name='linear_layer'),
    keras.layers.Dense(units=1, name='output_layer')
])

# 3. 컴파일 및 학습
# 선형 회귀에는 'adam' 옵티마이저와 'mse' 손실함수가 가장 적합합니다.
model.compile(optimizer=keras.optimizers.Adam(0.1), loss='mse')
print("학습 시작...")
model.fit(X, y, epochs=200, verbose=0)

# 4. 검증 (입력 10일 때 결과는 약 21이 나와야 함)
test_val = np.array([[10.0]])
prediction = model.predict(test_val)
print(f"예측 결과: 입력 10.0 -> 출력 {prediction[0][0]:.4f} (정답: 21.0)")

# 5. 세 가지 방식으로 저장 (확장 프로그램 변환 대비)
model.save('linear_reg_model.keras')          # Keras V3
model.save('linear_reg_model.h5')             # Legacy H5
model.save_weights('linear_reg_weights.weights.h5') # 가중치 전용

print("저장 완료: linear_reg_model.keras, .h5, .weights.h5")
```

Tensorflow 모델을 Javascript기반에서 실행할 수 있도록 변환하기

- 파이썬 기반에서 Tensorflowjs 모듈을 사용하여 모델 변환
  - 아래의 툴은 버전 차이로 오류를 발생할 경우가 많으므로 아래의 변환용 코드를 사용하는 것을 고려해야 함

```
# tensorflowjs_converter --input_format=keras [모델경로] [저장될디렉토리]
tensorflowjs_converter --input_format=keras ./odd_even_model.keras ./tfjs_model
```

- 변환용 코드를 사용하여 변환하기(*.keras, *.h5 모두 생성해서 테스트한다 (h5 포맷이 호환성이 더 높음))
- 아래의 코드에서 사용된 가중치만 로드하는 방식은 파일의 구조에 의존하지 않으므로 호환성이 더 높음

```
import sys
from types import ModuleType

# 1. Windows 호환성 가짜 모듈 주입
dummy_tfdf = ModuleType('tensorflow_decision_forests')
sys.modules['tensorflow_decision_forests'] = dummy_tfdf
sys.modules['tensorflow_decision_forests.keras'] = ModuleType('keras')

import tensorflow as tf
import tensorflowjs as tfjs

print("최종 해결책: load_weights 방식 시작")

# 2. 새 모델 정의 (학습 시 모델 구조와 정확히 동일해야 함)
new_model = tf.keras.Sequential([
    # 입력 1개, 출력 1개인 가장 단순한 선형 레이어
    tf.keras.layers.Dense(units=1, input_shape=(1,), name='linear_layer'),
    tf.keras.layers.Dense(units=1, name='output_layer')
])

try:
    # 3. 가중치 파일 로드 (.weights.h5 파일을 사용하면 메타데이터 충돌이 없습니다)
    # 만약 odd_even_weights.weights.h5가 없다면 odd_even_model.h5를 넣어보세요.
    try:
        new_model.load_weights('linear_reg_weights.weights.h5')  # linear_reg_weights.
        print("성공: 전용 가중치 파일에서 로드 완료")
    except:
        new_model.load_weights('linear_reg_model.h5')
        print("성공: 전체 모델 파일에서 가중치 로드 완료")

    # 4. TFJS 변환
    tfjs.converters.save_keras_model(new_model, 'tfjs_model')
    print("\n" + "="*50)
    print("변환 성공! ./tfjs_model 폴더를 확인하세요.")
    print("="*50)
```

```
except Exception as e:
    print(f"오류 발생: {e}")
```

- 변환된 모델 구성파일 확인 (위의 명령에서 사용한 tfjs_model 디렉토리 확인)

```
model.json: 모델의 구조(Layer, 가중치 연결 등)가 정의된 파일.
group1-shard1of1.bin: 실제 학습된 가중치(Weight) 데이터가 담긴 이진 파일
```

크롬 확장프로그램에 변환된 모델 추가하기

- model.json, group1-shard1of1.bin 파일 내용을 Javascript코드에 포함
- Javascript기반에서 사용되는 tensorflow.js 라이브러리 사용하여 모델 실행
- 브라우저 화면에 분류 결과 표시

1단계: 확장 프로그램 폴더 구조 만들기

- my_extension/
  - manifest.json (확장 프로그램 설정)
  - popup.html (사용자 화면)
  - popup.js (모델 로드 및 로직)
  - tf.min.js (TensorFlow.js 라이브러리)
  - tf-core.min.js
  - tf-backend-cpu.min.js
  - tf-layers.min.js
  - model/ (변환된 폴더)
    - model.json
    - group1-shard1of1.bin

2단계: TensorFlow.js 라이브러리 다운로드

- TensorFlow.js 공식 GitHub에서 tf.min.js 등을 다운로드하여 폴더에 저장(웹 검색)

3단계: manifest.json 작성

- 확장프로그램의 정보 정의 (web_accessible_resources 설정으로 브라우저가 .bin가중치 파일에 접근 가능함)

```
{
  "manifest_version": 3,
  "name": "y = 2x + 1 회귀 모델",
  "version": "1.0",
  "action": { "default_popup": "popup.html" },
```

```
  "web_accessible_resources": [
    {
      "resources": ["model/*", "*.wasm"],
      "matches": ["<all_urls>"]
    }
  ]
}
```

4단계: popup.html 작성

- 숫자를 입력하여 확장 프로그램에 입력할 웹 페이지
- https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@2.0.0/dist/tf.min.js 에 접속하여 tf.min.js 파일을 다운로드하거나 아래처럼...

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>y = 2x + 1 회귀 모델</title>
</head>
<body style="width: 200px; padding: 10px;">
  <h3>y = 2x + 1 회귀 모델</h3>
  <input type="number" id="numberInput" placeholder="숫자 입력">
  <button id="predictBtn">예측하기</button>
  <p id="resultText">모델 로딩 중...</p>

    <!-- popup.html -->
    <script src="tf-core.min.js"></script>
    <script src="tf-backend-cpu.min.js"></script>
    <script src="tf-layers.min.js"></script>
    <script src="popup.js"></script>

</body>
</html>
```

5단계: popup.js 작성

- 모델을 불러오고 예측을 실행하는 핵심 로직

```
// popup.js

// 전역 객체 확보 (tf-core 로드 확인)
const tfEngine = window.tf;

async function loadModel() {
  try {
    // 1. CPU 백엔드 명시적 설정
```

```
    await tfEngine.setBackend('cpu');
    console.log("CPU Backend 활성화");

    // 2. 모델 로드
    const modelUrl = chrome.runtime.getURL('model/model.json');

    // tf-layers가 로드되면 tf.loadLayersModel이 활성화됩니다.
    model = await tfEngine.loadLayersModel(modelUrl);

    document.getElementById('resultText').innerText = "AI 준비 완료!";
  } catch (error) {
    document.getElementById('resultText').innerText = "로드 실패: " + error.message;
    console.error("Detail Error:", error);
  }
}

// 예측 함수 내에서도 tfEngine(또는 window.tf)을 사용하세요.
async function predict() {
  const inputVal = document.getElementById('numberInput').value;
  if (!inputVal || !model) return;

  const num = parseFloat(inputVal);
  const inputTensor = tfEngine.tensor2d([[num]]);
  const prediction = model.predict(inputTensor);
  const y_val = (await prediction.data())[0];

  document.getElementById('resultText').innerText = `예측 결과: ${y_val}`;
}

document.getElementById('predictBtn').addEventListener('click', predict);
loadModel();
```

6단계: 크롬에 설치하기

1. 크롬 브라우저 주소창에 chrome://extensions/를 입력하여 접속.

2. 오른쪽 상단의 '개발자 모드'를 켭니다.

3. 왼쪽 상단의 '압축해제된 확장 프로그램을 로드' 버튼을 클릭.

4. 작업한 my_extension 폴더를 선택.

5. 크롬의 일반 웹브라우저 창으로 나가서 주소창 옆 퍼즐 아이콘 -> 방금 등록한 확장 프로그램 클릭 -> 고정핀 아이콘 클릭(항상 보이게 됨) -> 누르면 확장 프로그램이 실행

5. 확장 프로그램 아이콘을 클릭하여 숫자를 입력하고 결과를 확인

3. 크롬 브라우저에서 DevTools Extension 생성하기

- 일반 확장 프로그램과 달리 크롬의 개발자 도구를 사용해야 함
- 웹 응답 데이터에 대한 완전한 접근 허용
  - ✔ 압축 해제된 실제 응답 본문
  - ✔ JS / HTML / JSON / Text
  - ✔ 악성코드 분석에 필요한 원본 데이터
- 보안 분석 도구, 기업용 검사 시스템은 거의 전부 이 구조를 사용

3-1 DevTools Extension 구조 개요

```
Chrome
 └ DevTools
     └ Network Panel
         └ devtools_page (확장)
             └ JS로 네트워크 요청/응답 접근
```

3-2 기본 파일 구조

```
malware-devtools-extension/
├ manifest.json
├ devtools.html
├ devtools.js
└ (추후)
   ├ tfjs/
   ├ model.json
   └ classifier.js
```

3-3 manifest.json (DevTools Extension 선언)

- devtools_page 선언이 핵심

```
{
  "manifest_version": 3,
  "name": "DevTools Malware Scanner",
  "version": "1.0",
  "description": "Scan web responses for malicious code",
  "devtools_page": "devtools.html"
}
```

3-4 devtools.html

- UI선언도 가능하지만 처음에는 JS 로드만으로도 충분함

```html
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
</head>
<body>
  <script src="devtools.js"></script>
</body>
</html>
```

3-5 devtools.js

- chrome.devtools.network.onRequestFinished : 네트워크 요청이 완전히 끝난 후에 발생하는 이벤

```
chrome.devtools.network.onRequestFinished.addListener(
  (request) => {
    console.log("URL:", request.request.url);
    console.log("Method:", request.request.method);
    console.log("Status:", request.response.status);
    console.log("MimeType:", request.response.content.mimeType);

    request.getContent((body, encoding) => {
      console.log("Response body:", body);   // body: 문자열 (이미 디코딩됨)
      console.log("Encoding:", encoding);    // encoding: "base64" | null

      // 👉 여기에 악성코드 검사 로직 연결
    });
  }
);
```

3-6 악성코드 분석에 필요한 최소 필터링

- 실무에서는 모든 응답을 검사하지 않음

```
const ALLOW = [
  "javascript",
  "html",
  "json"
];
chrome.devtools.network.onRequestFinished.addListener(
  function (request) {
    try {
      const url = request.request.url;
      const mime = request.response.content.mimeType || "unknown";

      if (!ALLOW.some(type => mime.includes(type))) return;

      console.log("[SCAN]", url, mime);
```

```
      request.getContent(function (body) {
        if (!body) return;
        console.log(body.substring(0, 300));
        // 👉 ML 분석 대상
      });

    } catch (e) {
      console.error("DevTools error:", e);
    }
  }
);
```

3-7 확장 프로그램 등록 및 테스트

- chrome://extensions/ 접속
- 압축해제된 확장 프로그램 로드 -> "확장 프로그램 로드됨" 메시지 확인
- 다른 탭을 열고 > 주소창 3점 아이콘 클릭 > 도구 더보기 > 개발자 도구
- 개발자 도구가 열린 채로 원하는 웹사이트에 접속, 개발자 도구가 열리지 않은 상태로 접속하면 DevTools 확장 프로그램이 작동하지 않음
- chrome://extensions/ 접속 -> 해당 확장프로그램에서 "뷰 검사" 링크 클릭 > DevTools 창의 console 에서 메시지 출력 확인

4. DevTools 확장 프로그램에서 얻은 Response Body를 ML 입력을 위한 Feature로 변환하기

- 브라우저 확장 환경 → 경량 + 빠름이 최우선
- 정적 특징(Static Features) 중심
- 정적 + ML + 휴리스틱 혼합이 업계 표준
- 아래에 소개되는 모든 함수는 onRequestFinished 이벤트 핸들러와 동등한 레벨에 선언하고 request.getContent() 함수 안에서 호출되어야 함함

4-1 전체 그림

```
Response Body (JS / HTML / JSON / Text)
 └ Preprocessing
     ├ 정규화 (Normalization)
     ├ 길이 / 문자 통계
     ├ 키워드 / API 사용 패턴
     ├ 난독화 지표
     ├ 구조적 패턴
```

```
└ Feature Vector (숫자 배열)
   └ TensorFlow.js 모델 입력
```

4-2 전처리 (Preprocessing)

- 텍스트가 포함한 특성들을 Feature Vector로 생성 (Feature Vector는 모델에 입력할 수 있는 숫자로 구성됨)

4-2-1 기본 정규화 (필수)

- ✔ 난독화 차이 감소
- ✔ Feature 안정성 증가

```
function normalizeText(text) {
  return text
    .replace(/\s+/g, " ")   // 공백 정리
    .replace(/[^\x20-\x7E]/g, "") // 비ASCII 제거
    .toLowerCase();
}
```

4-2-2 길이 기반 Feature (가장 강력)

- 악성 JS는 비정상적으로 길거나 반대로 짧은 1줄 eval 형태가 많음

```
function lengthFeatures(text) {
  return {
    length: text.length,
    lineCount: text.split("\n").length,
    avgLineLength: text.length / Math.max(1, text.split("\n").length)
  };
}
```

4-2-3 문자 통계 기반 Feature (난독화 탐지 핵심)

- 문자 분포포

```
function charStats(text) {
  const total = text.length;
  let digits = 0, letters = 0, symbols = 0;

  for (const c of text) {
    if (/[0-9]/.test(c)) digits++;
    else if (/[a-z]/i.test(c)) letters++;
    else symbols++;
  }
```

```
  return {
    digitRatio: digits / total,
    letterRatio: letters / total,
    symbolRatio: symbols / total
  };
}
```

- 엔트로피(Entropy) (중요)
  - 정상 JS: 3.5 ~ 4.5
  - 난독화 JS: 5.0 이상

```
function entropy(text) {
  const freq = {};
  for (const c of text) freq[c] = (freq[c] || 0) + 1;

  let ent = 0;
  const len = text.length;

  for (const c in freq) {
    const p = freq[c] / len;
    ent -= p * Math.log2(p);
  }
  return ent;
}
```

4-2-4 악성 키워드 / API 패턴 Feature

- 고위험 키워드 카운트

```
const SUSPICIOUS_KEYWORDS = [
  "eval",
  "function(",
  "new function",
  "settimeout(",
  "setinterval(",
  "atob(",
  "fromcharcode",
  "document.write",
  "window.location",
  "unescape("
];

function keywordFeatures(text) {
  const features = {};
  for (const key of SUSPICIOUS_KEYWORDS) {
    const regex = new RegExp(key, "g");
    features[key] = (text.match(regex) || []).length;
  }
```

```
  return features;
}
```

- 네트워크 / C2 힌트

```
function networkHints(text) {
  return {
    hasHttp: /http:\/\//.test(text) ? 1 : 0,
    hasHttps: /https:\/\//.test(text) ? 1 : 0,
    hasIP: /\b\d{1,3}(\.\d{1,3}){3}\b/.test(text) ? 1 : 0
  };
}
```

4-2-5 구조적 Feature (JS / HTML 공통)

```
function structureFeatures(text) {
  return {
    scriptTagCount: (text.match(/<script/gi) || []).length,
    iframeCount: (text.match(/<iframe/gi) || []).length,
    base64Like: (text.match(/[A-Za-z0-9+/]{100,}={0,2}/g) || []).length
  };
}
```

4-2-6 Feature Vector로 변환 (ML 입력)

- TensorFlow.js는 숫자 배열을 요구함
- ✔ 고정 길이 벡터
- ✔ JS/HTML 공통 사용 가능
- ✔ 브라우저에서 빠름

```
function extractFeatures(rawText) {
  const text = normalizeText(rawText);

  const f1 = lengthFeatures(text);
  const f2 = charStats(text);
  const f3 = keywordFeatures(text);
  const f4 = networkHints(text);
  const f5 = structureFeatures(text);

  return [
    f1.length,
    f1.lineCount,
    f1.avgLineLength,
    f2.digitRatio,
    f2.letterRatio,
    f2.symbolRatio,
```

```
    entropy(text),
    ...Object.values(f3),
    f4.hasHttp,
    f4.hasHttps,
    f4.hasIP,
    f5.scriptTagCount,
    f5.iframeCount,
    f5.base64Like
  ];
}
```

4-2-7 DevTools Extension에서의 실제 연결 위치

```
request.getContent((body) => {
  if (!body || body.length < 50) return;

  const features = extractFeatures(body);

  console.log("Feature Vector:", features);

  // 다음 단계:
  // model.predict(tf.tensor([features]))
});
```

4-3 실제 devtools.js 예제 (실행 가능한 형태)

- devtools.js

```
// ================================
// Feature Extraction (GLOBAL)
// ================================

function normalizeText(text) {
  return text
    .replace(/\s+/g, " ")
    .replace(/[^\x20-\x7E]/g, "")
    .toLowerCase();
}

function lengthFeatures(text) {
  const lines = text.split("\n");
  return {
    length: text.length,
    lineCount: lines.length,
    avgLineLength: text.length / Math.max(1, lines.length)
  };
}
```

```
function charStats(text) {
  let digits = 0, letters = 0, symbols = 0;
  for (const c of text) {
    if (/[0-9]/.test(c)) digits++;
    else if (/[a-z]/i.test(c)) letters++;
    else symbols++;
  }
  const total = text.length || 1;
  return {
    digitRatio: digits / total,
    letterRatio: letters / total,
    symbolRatio: symbols / total
  };
}

function entropy(text) {
  const freq = {};
  for (const c of text) freq[c] = (freq[c] || 0) + 1;
  let ent = 0;
  const len = text.length || 1;
  for (const c in freq) {
    const p = freq[c] / len;
    ent -= p * Math.log2(p);
  }
  return ent;
}

const SUSPICIOUS_KEYWORDS = [
  "eval", "atob", "fromcharcode", "document.write",
  "settimeout", "setinterval", "unescape"
];

function keywordFeatures(text) {
  const result = {};
  for (const k of SUSPICIOUS_KEYWORDS) {
    result[k] = (text.match(new RegExp(k, "g")) || []).length;
  }
  return result;
}

function extractFeatures(rawText) {
  const text = normalizeText(rawText);
  const lf = lengthFeatures(text);
  const cs = charStats(text);
  const kw = keywordFeatures(text);

  return [
    lf.length,
    lf.lineCount,
    lf.avgLineLength,
    cs.digitRatio,
    cs.letterRatio,
    cs.symbolRatio,
```

```
    entropy(text),
    ...Object.values(kw)
  ];
}

// ================================
// DevTools Network Hook
// ================================

chrome.devtools.network.onRequestFinished.addListener(
  (request) => {
    request.getContent((body) => {
      if (!body || body.length < 50) return;

      const features = extractFeatures(body);
      console.log("[FEATURE VECTOR]", features);
      // 다음 단계:
      // model.predict(tf.tensor([features]))
    });
  }
);
```

4-3-1 위에서 선언한 DevTools 확장 프로그램 테스트

- 위의 확장 프로그램이 실행되면 DevTools 콘솔에 14차원 Feature Vector가 생성되어 표시됨
- 한개의 요청 안에서 네트워크 접속 시도 횟수만큼의 Feature Vector 수가 생성됨
- 컨텐츠가 길어도 네트워크 접속 1회마다 14차원 벡터 1개가 생성됨
- 요청 1개에서는 내부에서 많은 네트워크 접속이 이루어져서 컨텐츠를 다운로드하므로 다수개의 Feature Vector가 생성
- ✔️ 14차원은 → 지금까지 정의한 Feature 함수들이 만들어낸 결과의 개수이다.
- 사람이 이해할 수 있는 14개의 '의심 지표'를 동시에 보고 판단
- 👉 즉, DevTools 확장 프로그램에 최적화된 Feature Set

5. Feature Vector를 입력으로 받는 TensorFlow.js 모델 연결 및 추론

5-1 전체 흐름 한 눈에 보기

```
Network Response
 └ extractFeatures()  → [14]
    └ tf.tensor2d([features], [1,14])
       └ model.predict()
          └ 악성 확률 (0~1)
```

5-2 파일 구조 (권장)

- DevTools Extension에서는 CDN 사용을 권장하지 않음 (네트워크 차단, CSP 문제 방지)
- 함수들은 모두 features.js 파일에 선언

```
malware-devtools-extension/
├─ manifest.json
├─ devtools.html
├─ devtools.js          ← 네트워크 후킹 & 파이프라인
├─ features.js          ← Feature Extraction (14차원)
├─ model/
│   ├─ model.json
│   └─ group1-shard1of1.bin
└─ tf.min.js            ← TensorFlow.js (로컬 포함 권장)
```

5-2 devtools.html – 스크립트 로딩 순서 (매우 중요)

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
</head>
<body>
  <!-- TensorFlow.js -->
  <script src="tf.min.js"></script>

  <!-- Feature Extraction -->
  <script src="features.js"></script>

  <!-- DevTools Logic -->
  <script src="devtools.js"></script>
</body>
</html>
```

5-3 모델 준비 (사전 단계)

- 모델 조건
  - 입력 차원 : 14
  - 출력 : 1 (sigmoid)
- Python -> TF.js 변환
  - tensorflowjs_converter 툴이나 변환용 코드 사용
- 변환 결과:
  - model.json
  - group1-shared1of1.bin

5-4 devtools.js – 모델 로딩 (최초 1회)

- chrome.runtime.getURL() 필수

- DevTools 시작 시 한 번만 로드

```
let model = null;
let modelReady = false;

async function loadModel() {
  model = await tf.loadLayersModel(
    chrome.runtime.getURL("model/model.json")
  );
  modelReady = true;
  console.log("[ML] Model loaded");
}

loadModel();
```

5-5 추론함수 정의 (표준 패턴)

```
function predictMalicious(features) {
  if (!modelReady) return null;

  // [1, 14]
  const input = tf.tensor2d([features], [1, 14]);

  const output = model.predict(input);
  const score = output.dataSync()[0];

  // 메모리 정리 (필수)
  input.dispose();
  output.dispose();

  return score;
}
```

5-6 DevTools 네트워크 응답 → 추론 연결

```
chrome.devtools.network.onRequestFinished.addListener(
  (request) => {
    request.getContent((body) => {
      if (!body || body.length < 50) return;
      if (!modelReady) return;

      const features = extractFeatures(body);
      const score = predictMalicious(features);

      if (score !== null && score > 0.7) {
        console.warn("[MALICIOUS]", request.request.url, score);
      }
    });
```

```
  }
);
```

5-7 devtools.js 전체 예제 (실전용, 권장)

```
// ==========================================
// devtools.js
// DevTools Malware Scanner - ML Pipeline
// ==========================================

// ---- Global Model State ----
let model = null;
let modelReady = false;

// ---- 1. Load TensorFlow.js Model (ONCE) ----
async function loadModel() {
  try {
    const modelUrl = chrome.runtime.getURL("model/model.json");
    model = await tf.loadLayersModel(modelUrl);
    modelReady = true;
    console.log("[ML] Model loaded successfully");
  } catch (err) {
    console.error("[ML] Model loading failed:", err);
  }
}

// Load model immediately when DevTools opens
loadModel();

// ---- 2. Prediction Function ----
function predictMalicious(features) {
  if (!modelReady) return null;

  // Expecting features.length === 14
  const input = tf.tensor2d([features], [1, 14]);

  const output = model.predict(input);
  const score = output.dataSync()[0];

  // IMPORTANT: prevent memory leak
  input.dispose();
  output.dispose();

  return score;
}

// ---- 3. Network Response Hook ----
chrome.devtools.network.onRequestFinished.addListener(
  (request) => {
    request.getContent((body) => {
```

```
      // Basic guards
      if (!body || body.length < 50) return;
      if (!modelReady) return;

      // MIME-type filtering (recommended)
      const mime = request.response.content.mimeType || "";
      if (
        !mime.includes("javascript") &&
        !mime.includes("html") &&
        !mime.includes("json")
      ) {
        return;
      }

      try {
        // ---- Feature Extraction ----
        const features = extractFeatures(body);

        // ---- ML Inference ----
        const score = predictMalicious(features);

        if (score === null) return;

        // ---- Output ----
        if (score > 0.7) {
          console.warn(
            "[MALICIOUS]",
            request.request.url,
            "score:",
            score.toFixed(3)
          );
        } else {
          console.log(
            "[BENIGN]",
            request.request.url,
            "score:",
            score.toFixed(3)
          );
        }

      } catch (err) {
        console.error("[SCAN ERROR]", err);
      }
    });
  }
);
```

5-8 메모리 관리 (DevTools Extension에서 매우 중요)

- 메모리가 부족하면 DevTools가 느려지고 멈추게 된다

```
const input = tf.tensor2d([features], [1, 14]);
return model.predict(input);
input.dispose();
output.dispose();
```

- 또는 안전하게

```
tf.tidy(() => {
  const input = tf.tensor2d([features], [1, 14]);
  return model.predict(input);
});
```

5-9 정상 동작 체크 포인트

- ✔️ 모델 1회 로드
- ✔️ 네트워크 응답마다 score 출력
- ✔️ threshold 초과 시 경고

```
[ML] Model loaded
[MALICIOUS] https://example.com/script.js 0.83
```

5-10 여러 응답 점수를 하나의 페이지 Risk Score로 합치는 방법

- ✔️ 탐지력
- ✔️ 안정성
- ✔️ 설명 가능성
- ➡️ 실무에서 가장 많이 선택됨
- Hybrid (Max + Count) ⭐ ⭐ ⭐ (강력 추천)

```
pageRisk = max(score_i) * α + count_ratio * (1-α)
```

5-10-1 PageRisk 점수를 계산하는 DevTools Extension용 구현 예제

```
// ========================================================
// devtools.js
// DevTools Malware Scanner – Page-level Risk Aggregation
// ========================================================

// ------------------------------
// Global State
// ------------------------------
let model = null;
```

```
let modelReady = false;

// Page-level accumulated scores
let pageScores = [];

// ------------------------------
// 1. Load TensorFlow.js Model (ONCE)
// ------------------------------
async function loadModel() {
  try {
    const modelUrl = chrome.runtime.getURL("model/model.json");
    model = await tf.loadLayersModel(modelUrl);
    modelReady = true;
    console.log("[ML] Model loaded");
  } catch (err) {
    console.error("[ML] Failed to load model:", err);
  }
}

// Load model when DevTools opens
loadModel();

// ------------------------------
// 2. Single-response Prediction
// ------------------------------
function predictMalicious(features) {
  if (!modelReady) return null;

  // Expecting features.length === 14
  const input = tf.tensor2d([features], [1, 14]);
  const output = model.predict(input);
  const score = output.dataSync()[0];

  // Prevent memory leak
  input.dispose();
  output.dispose();

  return score;
}

// ------------------------------
// 3. Page-level Risk Aggregation
// ------------------------------
function computePageRisk(scores) {
  if (!scores || scores.length === 0) {
    return {
      pageRisk: 0,
      maxScore: 0,
      suspiciousCount: 0,
      total: 0
    };
  }
```

```
  const maxScore = Math.max(...scores);
  const threshold = 0.7;
  const suspiciousCount = scores.filter(s => s > threshold).length;
  const countRatio = suspiciousCount / scores.length;

  const alpha = 0.7; // weight for max score
  const pageRisk = alpha * maxScore + (1 - alpha) * countRatio;

  return {
    pageRisk,
    maxScore,
    suspiciousCount,
    total: scores.length
  };
}

// ------------------------------
// 4. Network Response Hook
// ------------------------------
chrome.devtools.network.onRequestFinished.addListener(
  (request) => {
    request.getContent((body) => {
      if (!body || body.length < 50) return;
      if (!modelReady) return;

      const mime = request.response.content.mimeType || "";
      if (
        !mime.includes("javascript") &&
        !mime.includes("html") &&
        !mime.includes("json")
      ) {
        return;
      }

      try {
        // ---- Feature Extraction ----
        const features = extractFeatures(body);

        // ---- ML Inference ----
        const score = predictMalicious(features);
        if (score === null) return;

        // ---- Accumulate page-level scores ----
        pageScores.push(score);

        // ---- Compute page risk ----
        const result = computePageRisk(pageScores);

        // ---- Output ----
        if (result.pageRisk > 0.8) {
          console.warn(
            "[PAGE HIGH RISK]",
            "risk:", result.pageRisk.toFixed(3),
```

```
            "| max:", result.maxScore.toFixed(3),
            "| suspicious:", result.suspiciousCount,
            "/", result.total
          );
        } else {
          console.log(
            "[PAGE STATUS]",
            "risk:", result.pageRisk.toFixed(3),
            "| max:", result.maxScore.toFixed(3),
            "| suspicious:", result.suspiciousCount,
            "/", result.total
          );
        }

      } catch (err) {
        console.error("[SCAN ERROR]", err);
      }
    });
  }
);

// ------------------------------
// 5. Reset on Page Navigation
// ------------------------------
chrome.devtools.network.onNavigated.addListener(() => {
  pageScores = [];
  console.log("[PAGE] Navigation detected – scores reset");
});
```

5-11 휴리스틱 룰 + ML 점수 결합 전략

6. 14차원의 Feature Vector를 이용한 Tensorflow 악성코드 분류모델 생성

6-1 Feature Vector → 모델 학습용 데이터셋 설계

- 실제 악성코드와 실제 정상 코드 그리고 모의 데이터로 구성해야 함
- 실제 악성 코드는 사람이 짐작하기 어렵게 되어 있거나 이미 알려진 방식을 회피하고 있음
- Feature Vector의 형태

```
{
  "features": [14 numbers],
  "label": 1
}
```

6-2 모의 데이터로 Feature Vector 생성하는 예

- 실제 Feature 분포를 모르면 위험
- 모델이 "가짜 경계"를 학습할 수 있음

```
function generateMockFeature(malicious=true) {
  if (malicious) {
    return [
      5000, 1, 5000,   // length, lines, avg
      0.4, 0.2, 0.4,   // char ratios
      5.8,             // entropy
      2,1,1,1,1,0,1    // keyword counts
    ];
  }
  return [
      800, 40, 20,
      0.1, 0.7, 0.2,
      3.9,
      0,0,0,0,1,0,0
  ];
}
```

6-3 현실적인 학습 전략 (권장)

✅ 단계 1: 구조 검증 (PoC)

- 소량 실제 데이터
- 모의 데이터
- 모델 구조 검증

✅ 단계 2: 실제 데이터 중심 학습

- 실제 정상 + 실제 악성
- Feature 분포 분석
- threshold 조정

✅ 단계 3: 운영 튜닝

- False Positive 로그 수집
- Feature 조정
- 재학습

6-4 모의 데이터를 사용한 전체 개념 검증(PoC, Proof of Concept)

6-4-1 전체 절차 요약 (한 눈에)

```
[1] 14차원 Feature 정의 확정
[2] 모의 Feature Vector 생성 (Python)
[3] TensorFlow 분류 모델 학습 (Python)
[4] TensorFlow.js 모델로 변환
[5] DevTools Extension에 모델 적용
[6] PoC 검증 포인트 확인
```

6-4-2 전제 조건 (지금 상태 점검)

- ✔ Feature 차원: 14
- ✔ labels:
  - 0 = 정상 (benign)
  - 1 = 악성 (malicious)
  - ✔ 모델 목적: 이진 분류

6-4-3 모의 Feature Vector 설계 원칙 (중요)

- ❗ "임의 난수"로 만들면 안
- 반드시 반영해야 할 것
  - 악성 / 정상의 분포 차이
  - Feature 간 상관관계
  - 현실적인 범위
- Feature 의미

Index	Feature	정상 경향	악성 경향
0	length	300~2000	1000~10000
1	lineCount	多	1~3
2	avgLineLength	낮음	매우 높음
3	digitRatio	낮음	높음
4	letterRatio	높음	낮음
5	symbolRatio	보통	높음
6	entropy	3.5~4.5	5.0~6.5
7~13	keyword count	거의 0	≥1

6-4-4 모의 데이터 생성 (Python)

- generate_mock_data.py

```
"""
Mock dataset generator for 14D malware feature vectors
"""

import numpy as np

np.random.seed(42)

NUM_SAMPLES = 2000

X = []
y = []

def benign_sample():
    return [
        np.random.randint(300, 2000),        # length
        np.random.randint(10, 80),            # lines
        np.random.uniform(10, 60),            # avg line length
        np.random.uniform(0.05, 0.15),        # digitRatio
        np.random.uniform(0.6, 0.8),          # letterRatio
        np.random.uniform(0.1, 0.25),         # symbolRatio
        np.random.uniform(3.5, 4.5),          # entropy
        *np.random.binomial(1, 0.05, 7)       # keyword counts
    ]

def malicious_sample():
    return [
        np.random.randint(1000, 10000),
        np.random.randint(1, 5),
        np.random.uniform(200, 2000),
        np.random.uniform(0.25, 0.45),
        np.random.uniform(0.1, 0.3),
        np.random.uniform(0.3, 0.6),
        np.random.uniform(5.0, 6.5),
        *np.random.binomial(3, 0.6, 7)
    ]

for _ in range(NUM_SAMPLES // 2):
    X.append(benign_sample())
    y.append(0)

for _ in range(NUM_SAMPLES // 2):
    X.append(malicious_sample())
    y.append(1)

X = np.array(X, dtype=np.float32)
y = np.array(y, dtype=np.float32)
```

6-4-5 TensorFlow 모델 학습 (Python)

- train_model.py

```
import tensorflow as tf
from sklearn.model_selection import train_test_split

X_train, X_val, y_train, y_val = train_test_split(
    X, y, test_size=0.2, random_state=42
)

model = tf.keras.Sequential([
    tf.keras.layers.Input(shape=(14,)),
    tf.keras.layers.Dense(32, activation="relu"),
    tf.keras.layers.Dense(16, activation="relu"),
    tf.keras.layers.Dense(1, activation="sigmoid")
])

model.compile(
    optimizer="adam",
    loss="binary_crossentropy",
    metrics=["accuracy"]
)

model.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    epochs=20,
    batch_size=32
)

model.save("saved_model")
```

6-4-6 TensorFlow.js 포맷으로 모델 변환

- 변환용 툴(tensorflowjs_converter)을 사용하거나 변환용 코드를 사용하는 방법이 있음
- tensorflowjs 모듈에 포함된 tensorflowjs_converter는 버전에 따라 호환이 안되는 오류가 흔함
- 변환된 결과 아래의 파일이 생성됨
  - model.json
  - group1-shard1of1.bin

6-4-7 DevTools Extension에 적용

- 이미 구성된 구조를 그대로 사용

```
malware-devtools-extension/
├─ devtools.js
├─ features.js
├─ tf.min.js
└─ model/
    ├─ model.json
    └─ group1-shard1of1.bin
```

- 확장 프로그램 로드 후에 DevTools에서 확인

```
[MALICIOUS] https://example.com/script.js score: 0.82
```