**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**ĐỒ ÁN HỌC PHẦN**

TÊN HỌC PHẦN: **HỌC MÁY (Machine Learning: ML = Máy học)**

MÃ SỐ LỚP HP: **MALE431984\_01**

Tên đề tài: **PHÁT TRIỂN HỆ THỐNG HỌC MÁY NHẬN DIỆN MỘT SỐ CHỮ CÁI TIẾNG NHẬT VIẾT TAY DÙNG GIẢI THUẬT CNN**

**Họ tên sinh viên: Trần Ngọc Minh Thiện**

**Mã số sinh viên: 18110371**

**Lớp: 181102B**

**Ngày nộp: ……./……./202…**

**Ký tên:**

**TP.HCM, ngày …. Tháng 12 năm 2020**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

**ĐỀ TÀI ĐỒ ÁN HỌC PHẦN**

**Tên HP: HỌC MÁY Mã lớp HP:** MALE431984\_01

Tên đề tài: **PHÁT TRIỂN HỆ THỐNG HỌC MÁY NHẬN DIỆN MỘT SỐ CHỮ CÁI TIẾNG NHẬT VIẾT TAY DÙNG GIẢI THUẬT CNN**

**Giảng viên giảng dạy: VÕ XUÂN THỂ**

**Sinh viên thực hiện: Trần Ngọc Minh Thiện**

**Mã số sinh viên:** 18110371

**Lớp:** 181102B

**Tên sản phẩm đề tài: Katakana Recognizor**

**Công cụ sử dụng (phiên bản): Anaconda Navigator 1.9.12 và Spyder 4.1.4**

**Ngôn ngữ lập trình sử dụng: Python 3.7**

**Nhận xét của giảng viên:**

**………………………………………………………..**

**………………………………………………………..**

**………………………………………………………..**

**Điểm đánh giá:………….(…………………)**

**Ngày……./……../202…..**

**Giảng viên:**

**Ký tên**

**TP.HCM, ngày ……. tháng …… năm 202…….**

**LỜI CẢM ƠN**

Em xin gửi lời cám ơn chân thành đến thầy Võ Xuân Thể, giảng viên hướng dẫn môn Học máy – người đã mang lại những tiết học thú vị, những kiến thức bổ ích cho sinh viên. Chính nhờ những chỉ dạy tận tình của thầy đã giúp em hoàn thành được đồ án.

**DANH MỤC CÁC TỪ VIẾT TẮT**

|  |  |
| --- | --- |
|  | |
| AI | Artificial Intelligence: Trí tuệ nhân tạo |
| ANN | Artificial Neural Network: Mạng nơ-ron nhân tạo |
| CNN | Convolutional Neural Network: Mạng nơ-ron tích chập |
| ML | Machine Learning: Học máy = Máy học |

**DANH MỤC CÁC THUẬT NGỮ**

|  |  |
| --- | --- |
| Hồi quy tuyến tính | Là một phương pháp phân tích quan hệ giữa biến phụ thuộc Y với một hay nhiều biến độc lập X. |
| Lực | Là bất kỳ ảnh hưởng nào làm một vật thể chịu sự thay đổi, hoặc là ảnh hưởng đến chuyển động, hướng của nó hay cấu trúc hình học của nó |
| Quán tính | Quán tính là tính chất giữ nguyên chuyển động của một vật khi không có lực tác dụng và chỉ thay đổi dần chuyển động nếu có lực tác dụng |

**MỤC LỤC**

[**LỜI CẢM ƠN** 3](#_Toc61837632)

[**DANH MỤC CÁC TỪ VIẾT TẮT** 4](#_Toc61837633)

[**DANH MỤC CÁC THUẬT NGỮ** 5](#_Toc61837634)

[**PHIẾU GIAO NHIỆM VỤ ĐỀ TÀI ĐỒ ÁN HỌC PHẦN** 8](#_Toc61837635)

[Chương 1: GIỚI THIỆU VỀ ĐỀ TÀI 9](#_Toc61837636)

[Chương 2: CƠ SỞ LÝ THUYẾT VỀ HỌC MÁY 13](#_Toc61837637)

[**1.1.** **Chi tiết một số cơ sở lý thuyết:** 13](#_Toc61837638)

[**1.1.1.** **Căn bản về “học máy”: ML** 13](#_Toc61837639)

[**1.1.2.** **Các giải thuật ML có giám sát (có hướng dẫn)** 13](#_Toc61837640)

[**1.1.3.** **Các giải thuật ML không giám sát (không hướng dẫn)** 13](#_Toc61837641)

[**1.1.4.** **Python và các thư viện ML trên Python** 13](#_Toc61837642)

[**1.1.5.** **Giải thuật Hồi quy tuyến tính và ứng dụng** 14](#_Toc61837643)

[**1.1.6.** **Giải thuật Phân lớp theo láng giềng gần nhất kNNC và ứng dụng** 14](#_Toc61837644)

[**1.1.7.** **Giải thuật Mạng Nơ-ron nhân tạo lan truyền ngược BP\_ANN và ứng dụng** 14](#_Toc61837645)

[**1.1.8.** **Học không giám sát với giải thuật phân cụm (Clustering) k-means và ứng dụng** 14](#_Toc61837646)

[1.2. Chi tiết về giải thuật được sử dụng trong đề tài: BP\_ANN 14](#_Toc61837647)

[Chương 3: GIỚI THIỆU VỀ GIẢI THUẬT ML: ANN 16](#_Toc61837648)

[**3.1. Cấu trúc (hoạt động) của một Nơ-ron** 16](#_Toc61837649)

[**3.1.1. Đầu vào tổng thể (Net Input)** 16](#_Toc61837650)

[**3.1.2. Hàm tác động (Activation function)** 16](#_Toc61837651)

[**3.2. Kiến trúc của ANN** 18](#_Toc61837652)

[**3.3. Các quy tắc “học” của ANN** 19](#_Toc61837653)

[**3.3.1. Quy tắc “học” bằng trọng số tổng quát** 19](#_Toc61837654)

[**3.3.2. Perceptron** 20](#_Toc61837655)

[**3.3.3. Giải thuật Perceptron** 21](#_Toc61837656)

[**3.3.4. Vấn đề “hội tụ” của giải thuật Perceptron** 22](#_Toc61837657)

[**3.3.5. Hàm đánh giá lỗi (Error Function)** 22](#_Toc61837658)

[**3.3.6. Giải thuật của chiến lược Gradient Descent (giảm độ “dốc”)** 23](#_Toc61837659)

[**3.3.7. Giải thuật điều chỉnh trọng số adam** 24](#_Toc61837660)

[3.4. Giới thiệu về thuật toán Convolution Neural Network 26](#_Toc61837661)

[3.4.1 Giới thiệu đôi nét 26](#_Toc61837662)

[3.4.2. Lớp tích chập - Convolution Layer 26](#_Toc61837663)

[3.4.3. Bước nhảy - Stride 26](#_Toc61837664)

[3.4.4. Đường viền - Padding 26](#_Toc61837665)

[3.4.5. Hàm phi tuyến - ReLU 26](#_Toc61837666)

[3.4.6. Lớp gộp - Pooling Layer 26](#_Toc61837667)

[3.4.7. Tóm tắt 26](#_Toc61837668)

[Chương 4: GIỚI THIỆU: CÀI ĐẶT VÀ THỰC NGHIỆM HỆ THỐNG ML: ANN 28](#_Toc61837669)

[**4.1. Sơ lược về hệ thống** 28](#_Toc61837670)

[**4.1.1. Cấu trúc của ANN** 28](#_Toc61837671)

[**4.1.2. Hàm kích hoạt** 28](#_Toc61837672)

[4.2. **Code cài đặt** 29](#_Toc61837673)

[4.2.1. Code trích xuất dữ liệu từ tập tin ETL1 tạo file Extention **katakana.npz** 29](#_Toc61837674)

[4.2.2. Code định dạng kích thước toàn bộ tập ảnh, phân chia thành cái tập học, kiểm tra. 29](#_Toc61837675)

[4.2.3. Code train model 30](#_Toc61837676)

[4.2.4. Code tạo Form giao diện sử dụng tkinter 32](#_Toc61837677)

[4.2.5. Labels 34](#_Toc61837678)

[Chương 5: KẾT QUẢ THỰC NGHIỆM HỆ THỐNG ML: CNN 35](#_Toc61837679)

[Chương 6: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN CỦA ĐỀ TÀI 36](#_Toc61837680)

[**6.1.** **Kết luận** 36](#_Toc61837681)

[6.1.1. Những kết quả đạt được 36](#_Toc61837682)

[6.1.2. Hạn chế 36](#_Toc61837683)

[**6.2.** **Hướng phát triển** 36](#_Toc61837684)

[6.2.1. Hướng khắc phục các hạn chế 36](#_Toc61837685)

[6.2.2. Hướng mở rộng đề tài 36](#_Toc61837686)

[DANH MỤC TÀI LIỆU THAM KHẢO 37](#_Toc61837687)

[ĐÁNH GIÁ KẾT QUẢ HỌC TẬP HỌC PHẦN 38](#_Toc61837688)

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

**PHIẾU GIAO NHIỆM VỤ ĐỀ TÀI ĐỒ ÁN HỌC PHẦN**

|  |  |
| --- | --- |
| **Giảng viên: VÕ XUÂN THỂ** | |
| **Tên học phần:** Học máy **Mã lớp HP:** MALE431984\_01 | |
| **Tên đề tài:** Phát triển hệ thống học máy nhận diện một số chữ cái tiếng Nhật viết tay dùng giải thuật CNN  + Tên đề tài của Nhóm: Ứng dụng giải thuật BP\_ANN  + Phần chức năng của cá nhân: Ứng dụng CNN trong nhận diện một số hình vẽ tay | |
| **Sinh viên thực hiện:**   * **Họ tên:** Trần Ngọc Minh Thiện * **MSSV:** 18110371 | |
| **Thời gian thực hiện: 28/09/2020**  đến **05/01/2021** | |
| **Yêu cầu của đề tài**  **Lý thuyết:**   1. Căn bản về “học máy”: ML 2. Các giải thuật ML có giám sát (có hướng dẫn) 3. Các giải thuật ML không giám sát (không hướng dẫn) 4. Python và các thư viện ML trên Python 5. Giải thuật Hồi quy tuyến tính và ứng dụng 6. Giải thuật Phân lớp theo láng giềng gần nhất kNNC và ứng dụng 7. Giải thuật Mạng Nơ-ron nhân tạo lan truyền ngược BP\_ANN và ứng dụng 8. Học không giám sát với giải thuật phân cụm (Clustering) k-means và ứng dụng 9. Các giải thuật ML khác   Nhóm giải thuật chính được sử dụng trong đề tài: BP\_ANN (mục số 7)  **Thực hành:** Tên đề tài của Nhóm: Phát triển hệ thống học máy dùng giải thuật BP\_ANN  + Phần chức năng của cá nhân: Thư viện chính được dùng? Tập học là gì? Tập thực nghiệm là gì? Nội dung chính của giải thuật được dùng,….. | |
| **GIẢNG VIÊN** | **Ngày … tháng … năm 202….**  **SV Thực hiện** |

# GIỚI THIỆU VỀ ĐỀ TÀI

* 1. **Tổng quan về đề tài**

Học Máy (hoặc Máy Học, ML - Machine Learning) là một lĩnh vực chuyên sâu thuộc về Trí tuệ nhân tạo (AI - Artificial Intelligence). Các thuật toán ML là các chương trình máy tính có khả năng học hỏi về cách hoàn thành các nhiệm vụ và cách cải thiện hiệu suất theo thời gian.[[1]](#footnote-1)

* Đề tài tập trung xây dựng một hệ thống nhận dạng một số hình vẽ tay đơn giản.
* Giải thuật chính: ANN
* Thư viện: numpy, sklearn, keras, matplotlib, tkinter, tensorflow, PIL
* Tập dữ liệu: Các dữ liệu được lấy bộ dữ liệu ETL Character Database bao gồm 14 file trong thư mục zip Katakana(ETL 1).
* Thông tin về tập dữ liệu:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Filename | Categories | # Categories | Sheets | # Sheets | # Records |
| ETL1C-01 | 01234567 | 8 | 1001-2960 | 1445 | 11560 |
| ETL1C-02 | 89ABCDEF | 8 | 1001-2960 | 1445 | 11560 |
| ETL1C-03 | GHIJKLMN | 8 | 1001-2960 | 1445 | 11560 |
| ETL1C-04 | OPQRSTUV | 8 | 1001-2960 | 1445 | 11560 |
| ETL1C-05 | WXYZ\+-\* | 8 | 1001-2960 | 1445 | 11560 |
| ETL1C-06 | /=()・,?’ | 8 | 1001-2960 | 1445 | 11560 |
| ETL1C-07 | アイウエオカキク | 8 | 1001-2960 | 1411 | 11288 |
| ETL1C-08 | ケコサシスセソタ | 8 | 1001-2960 | 1411 | 11288 |
| ETL1C-09 | チツテトナニヌネ | 8 | 1001-2960 | 1411 | 11287 note: ナ(NA) on Sheet 2672 is missing |
| ETL1C-10 | ノハヒフヘホマミ | 8 | 1001-2960 | 1411 | 11288 |
| ETL1C-11 | ムメモヤイユエヨ | 8 | 1001-2960 | 1411 | 11288 |
| ETL1C-12 | ラリルレロワヰウ | 8 | 1001-2960 | 1411 | 11287 note: リ(RI) on Sheet 2708 is missing |
| ETL1C-13 | ヱヲン | 3 | 1001-2960 | 1411 | 4233 |

+ Link tải bộ dữ liệu : [etlcdb (aist.go.jp)](http://etlcdb.db.aist.go.jp/)

+ Trích xuất 7 tập tin chứa các ký tự katakana tạo thành File Extention **katakana.npz (**ETL1C-07 - ETL1C-013)

**+** Sau đó, định dạng toàn bộ hình ảnh từ file **katakana.npz** theo kích thước 48\*48, tạo nhãn cho từng ký tự, tách hình ảnh và nhãn thành các file huấn luyện và kiểm tra.

* Dữ liệu được phân lần lượt vào các tập tin tương ứng:

|  |  |
| --- | --- |
| **Số lượng** | **Minh họa** |
| katakana\_train\_images.npz | |
|  |  |
| katakana\_train\_labels.npz | |
|  |  |
| katakana\_test\_images.npz | |
|  |  |
| katakana\_test\_labels.npz | |
|  |  |

* 1. **Nội dung chuyên môn chính của đề tài**

+ Giải thuật: dùng mạng nơ-ron nhân tạo (ANN), xây dựng hệ thống để “học” từ tập dữ liệu.

* ANN: hệ thống mô phỏng hệ thống nơ-ron của con người

Cấu trúc ANN trong hệ thống: chia làm 3 tầng: tầng đầu vào, tầng ẩn, tầng đầu ra.

Cách học thông qua quá trình điều chỉnh trọng số cho các liên kết giữa các nơ-ron.

+ Thực nghiệm và kho dữ liệu sử dụng: Tập học: chiếm 80%, tập thực nghiệm: chiếm 20% của bộ dữ liệu

* 1. **Mục tiêu của đề tài**

+ Về công nghệ: vận dụng giải thuật CNN trong thực tiễn

+ Về thực tiễn ứng dụng: dùng trong nhận dạng vật thể, các trò chơi tương tác người dùng (qua camera)…

* 1. **Bố cục của báo cáo**

Chương 1 Giới thiệu tổng quan về đề tài

Chương 2 Cơ sở lý thuyết của học phần: gồm những lý thuyết cơ bản về ML liên quan đến đề tài.

Chương 3 Giới thiệu về Giải thuật ML: BP\_ANN và CNN

Chương 4 Giới thiệu về cài đặt và thực nghiệm hệ thống ML: CNN

Chương 5 Kết quản thực nghiệm hệ thống ML: CNN

Chương 6 Kết luận về kết quản đạt được và những tồn tại, trên cơ cở đó đề xuất các giải pháp khắc phục tồn và hướng mở rộng đề tài.

# CƠ SỞ LÝ THUYẾT VỀ HỌC MÁY

* 1. **Chi tiết một số cơ sở lý thuyết:**
     1. **Căn bản về “học máy”: ML**

Học máy là một lĩnh vực chuyên sâu của AI tập trung phát triển các hệ thống có khả năng “học”. Có hai cách phổ biến phân nhóm các giải thuật Machine learning:

**\* Dựa trên phương thức học (learning style): 6 loại chia trong 4 nhóm**

* Học có giám sát (Supervised Learning)
* Học không giám sát (Unsupervised Learning)
* Học bán giám sát (Semi-Supervised Learning)
* Học Tăng cường || củng cố (Reinforcement Learning)

**\* Dựa trên chức năng (function) của mỗi giải thuật: 9 loại**

* Giải thuật hồi quy = Regression Algorithms
* Giải thuật phân lớp || loại = Classification Algorithms
* Giải thuật dựa trên Mẫu = Instance-based Algorithms
* Giải thuật chuẩn hóa = Regularization Algorithms
* Giải thuật Bayes = Bayesian Algorithms
* Giải thuật phân cụm | nhóm = Clustering Algorithms
* Giải thuật Mạng Nơ-ron nhân tạo = Artificial Neural Network Algorithms
* Giải thuật giảm số chiều (kích thước) dữ liệu = Dimensionality Reduction Algorithms
* Giải thuật tổng hợp = Ensemble Algorithms (thông qua “bỏ phiếu” từ nhiều mô hình con yếu hơn = weaker model)
  + 1. **Các giải thuật ML có giám sát (có hướng dẫn)**
* Học có giám sát = Supervised learning là các giải thuật dự đoán đầu ra (outcome) của một dữ liệu đầu vào mới (new input) dựa vào tập các cặp (input, outcome) đã biết [đã được “học”]
* Cặp (input, outcome) đã biết có dạng (data, label) = (dữ liệu, nhãn).
* Phân loại:
  + Dùng phương pháp “Phân lớp | loại”: Classification
  + Dùng phương pháp “Hồi quy”: Regression. . .
    1. **Các giải thuật ML không giám sát (không hướng dẫn)**
* Với các Giải thuật “học KHÔNG giám sát”: chúng ta KHÔNG biết được outcome (tức label = nhãn) mà chỉ có dữ liệu đầu vào (input) mà dựa vào cấu trúc của dữ liệu để thực hiện một công việc nào đó.
* Phân loại:
  + Dùng phương pháp “Phân cụm | nhóm”: Clustering
  + Dùng phương pháp “Luật kết hợp”: Association
    1. **Python và các thư viện ML trên Python**
* Python là ngôn ngữ lập trình vừa cho phép “biên dịch” vừa cho phép “thông dịch”. Việc nhóm cấu trúc các câu lệnh thông qua canh lề (Indentation) và KHÔNG dùng ; kết thúc câu lệnh, KHÔNG cần khai báo trước các “biến” hay “tham số”. Python cho phép chia chương trình thành các module để tái sử dụng.
  + Các thư viện python thường dùng cho học máy: Pandas, Tensorflow, NumPy, Theano, Matplotlib, sklearn…
    1. **Giải thuật Hồi quy tuyến tính và ứng dụng**
* Thuật toán hồi quy tuyến tính thuộc vào nhóm học có giám sát, nhằm dự đoán giá trị của một hoặc nhiều biến mục tiêu liên tục y dựa trên một véc-to đầu vào x. Hàm hồi quy tuyến tính có dạng:

Trong đó: y là giá trị thực của outcome (dựa trên số liệu thống kê chúng ta có trong tập training data), là giá trị mà mô hình Linear Regression dự đoán được.

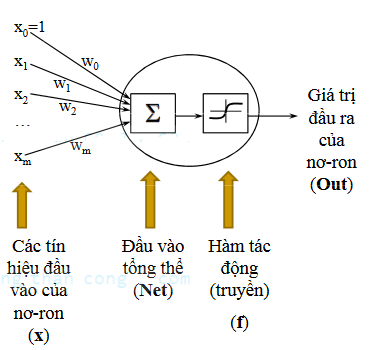
* Ứng dụng: dự đoán giá cổ phiếu, dự đoán giá nhà,…
  + 1. **Giải thuật Phân lớp theo láng giềng gần nhất kNNC và ứng dụng**
* Hàm mục tiêu có giá trị rời rạc đã được gán "nhãn", Output là một giá trị ("nhãn") cụ thể trong tập các giá trị rời rạc đã xác định trước (đã được gán "nhãn").
* Các giai đoạn hoạt động: [1] Giai đoạn “học”; [2] Giai đoạn “phân lớp” = thực nghiệm (Test)
* Ứng dụng: phân loại chữ viết tay, dự đoán phá sản, dự đoán giá cổ phiếu, phân bổ xếp hạng tín dụng trái phiếu doanh nghiệp, tạo ra chỉ số vốn và trái phiếu tùy chỉnh…
  + 1. **Giải thuật Mạng Nơ-ron nhân tạo lan truyền ngược BP\_ANN và ứng dụng**
* Máy học mô phỏng Mạng Nơ-ron nhân tạo (Artificial Neural Network – ANN) là hệ thống máy học mô phỏng các hệ thống nơ-ron sinh học (trong bộ não con người).
* Có 2 kiểu “học” trong ANN: “Học” bằng “Tham số” (Parameter ML), “Học” thay đổi cấu trúc (Structure ML).
* Ứng dụng: Xử lý ảnh và Computer vision, Xử lý tín hiệu, Nhận dạng mẫu, …
  + 1. **Học không giám sát với giải thuật phân cụm (Clustering) k-means và ứng dụng**
* Trong thuật toán phân cụm K-means clustering, chúng ta không biết nhãn (label) của từng điểm dữ liệu. Mục đích là làm thể nào để phân dữ liệu thành các cụm (cluster) khác nhau sao cho dữ liệu trong cùng một cụm có tính chất giống nhau.
* Ứng dụng: Xác định các nhóm khách hàng, phân nhóm động vật, phân loại tài liệu, phân loại người dùng web…

## Chi tiết về giải thuật được sử dụng trong đề tài: BP\_ANN

* ANN có cấu trúc dựa trên một số lượng các “nơ-ron” (Artificial Neurons) liên kết với nhau. Mỗi nơ-ron bao gồm:
  + Một giá trị đặc trưng vào [Input] | ra [Output]
  + Thực hiện một tính toán cục bộ tại “Nơ-ron” đó (một hàm cục bộ)
* Giá trị đặc trưng đầu ra của một “nơ-ron” được xác định bởi:
  + Giá trị đặc trưng vào | ra cụ thể của nó
  + Các liên kết của nó [Nơ-ron đó] với các Nơ-ron khác
  + (Có thể có thêm) các đầu vào bổ sung
* ANN có thể được xem là một cấu trúc xử lý thông tin vừa phân tán vừa song song. ANN có khả năng học (learn), nhớ lại (recall), và khái quát hóa (generalize) từ các dữ liệu học: bằng cách gán và điều chỉnh (tự thích nghi) các giá trị trọng số (hệ số đánh giá mức độ quan trọng) của các liên kết giữa các nơ-ron với nhau.
* Hàm mục tiêu [Chức năng] của một ANN được xác định bởi:
  + Kiến trúc (topology) của mạng nơ-ron
  + Đặc trưng vào | ra của mỗi nơron
  + Chiến lược học (huấn luyện)
  + Dữ liệu học
* Các ứng dụng điển hình của ANN:
  + Xử lý ảnh và Computer vision: Sao trùng, tiền xử lý, phân đoạn và phân tích ảnh computer vision, nén ảnh, xử lý và hiểu các ảnh thay đổi theo thời gian…
  + Xử lý tín hiệu: Phân tích tín hiệu và hình thái địa chấn, động đất…
  + Nhận dạng mẫu: Trích chọn thuộc tính, phân loại và phân tích tín hiệu ra đa, nhận dạng và hiểu giọng nói, nhận dạng dấu vân tay, nhận dạng ký tự (chữ hoặc số), nhận dạng mặt người, và phân tích chữ viết tay…
  + Y tế: Phân tích và hiểu tín hiệu điện tim, chẩn đoán các loại bệnh, xử lý các ảnh trong lĩnh vực y tế…
  + Và nhiều ứng dụng khác trong thực tế…

# GIỚI THIỆU VỀ GIẢI THUẬT ML: ANN và CNN

* 1. **Cấu trúc (hoạt động) của một Nơ-ron**



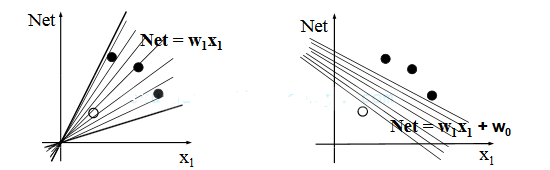
+ Các tín hiệu đầu vào (Input Signal) của Nơ-ron: xi (với i = 1,…,m). Mỗi tín hiệu đầu vào xi gắn liền với giá trị trọng số wi. Ngoài ra, còn có trọng số điều chỉnh (độ lệch: bias) = tín hiệu dịch chuyển w0.

+ Đầu vào tổng thể (Net Input): Là hàm tích hợp các tín hiệu đầu vào: Net(w, x)

+ Hàm tác động [lan truyền] [Activation || Transfer Function]: tính giá trị đầu ra của Nơ-ron: f(Net(w, x))

+ Giá trị đầu ra của Nơ-ron: Out= f(Net(w, x))

* + 1. **Đầu vào tổng thể (Net Input)**
* Thường được tính bởi 1 hàm tuyến tính:
* Ý nghĩa độ lệnh Bias = tín hiệu dịch chuyển: w0 dùng để tách Net thành 2 lớp (Classes): w0 = 0 và w0 != 0



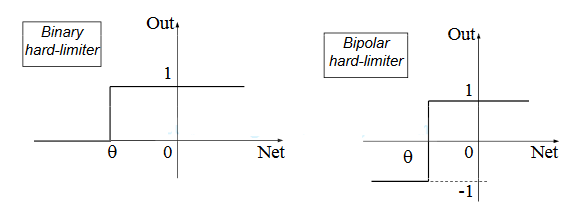
* + 1. **Hàm tác động (Activation function)**
* *Dạng giới hạn cứng [Hard-Limiter]*

Còn họi là hàm ngưỡng (Threshold function):

Giá trị đầu ra sẽ lấy 1 trong 2 giá trị nêu trên. Có thể dùng -1 [Bipolar Hard-Limter] thay cho 0 [Binary Hard-Limter].

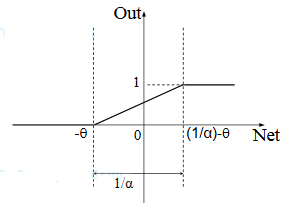
Nhược điểm: hàm không liên tục & đạo hàm không liên tục:

Out(Net) = hl(Net, θ) = sign(Net, θ)



* *Dạng Logic ngưỡng [Threshold Logic Function]*

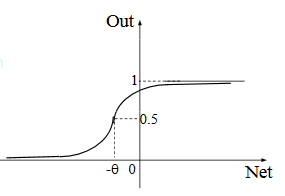
Còn gọi là hàm tuyến tính bão hòa (Saturating Linear Function).

Với , xác định độ “dốc” của khoảng tuyến tính.

Dễ thấy, đây là biến thể của giới hạn cứng [chặt] nêu trên = chèn giữa hàm tuyến tính với độ “dốc” .

Nhược điểm: hàm liên tục nhưng đạo hàm không liên tục

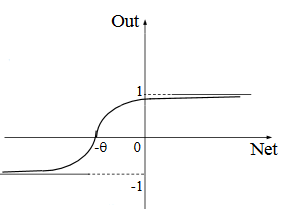
* *Dạng Xích-ma [Sigmoidal Function]*



Giá trị đầu ra trong khoảng (0…1). Được sử dụng phổ biến

Ưu điểm: hàm liên tục & đạo hàm liên tục (vì đạo hàm của hàm Xích-ma ra chính nó)

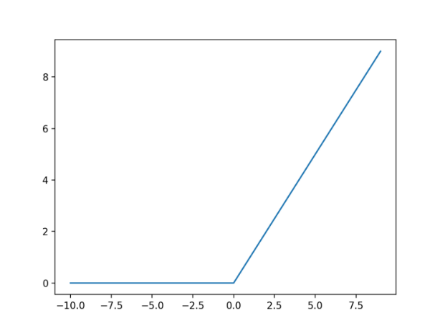
* *Dạng hàm tiếp tuyến Hyperbolic [Hyperbolic Tangent Function]*



Giá trị đầu ra trong khoảng (-1…1)

Cũng được sử dụng phổ biến

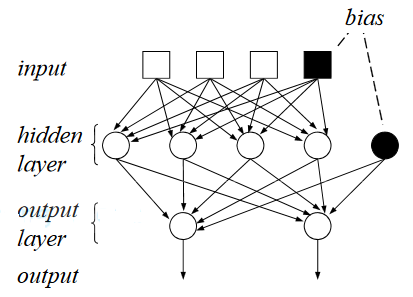
Ưu điểm: hàm liên tục & đạo hàm liên tục (vì đạo hàm của hàm tanh cũng ra chính nó)

* *Hàm kích hoạt ReLU - Rectified Linear Unit*

ReLU đơn giản lọc các giá trị < 0. Giá trị đầu ra (0…)

Hàm ReLU đang được sử dụng khá nhiều trong những năm gần đây khi huấn luyện các mạng neuron.

* 1. **Kiến trúc của ANN**
* Kiến trúc ANN được xác định bởi:
  + Số lượng tín hiệu Input || Output
  + Số lượng tầng
  + Số lượng Nơ-ron trong mỗi tầng
  + Số lượng các trọng số (tương ứng các liên kết Nơ-ron khác) của mỗi Nơ-ron
  + Cách thức các Nơ-ron trong mỗi tầng hoặc giữa các tầng liên kết nhau
  + Các Nơ-ron nhận tín hiệu điều chỉnh lỗi
* Mỗi ANN phải có:
  + Một tầng đầu vào (Input Layer)
  + Một tầng đầu ra (Output Layer)
  + Không -> nhiều Tầng ẩn (Hidden Layer[s]); thường có ít nhất 1

Ví dụ ANN hình bên có:

+ 3 tín hiệu Input (và 1 bias)

+ 2 tín hiệu Output

+ 6 Nơ-rons (và 1 bias) trong 3 tầng

4 Nơ-rons ở tầng Ẩn & 2 Nơ-ron ở tầng Output

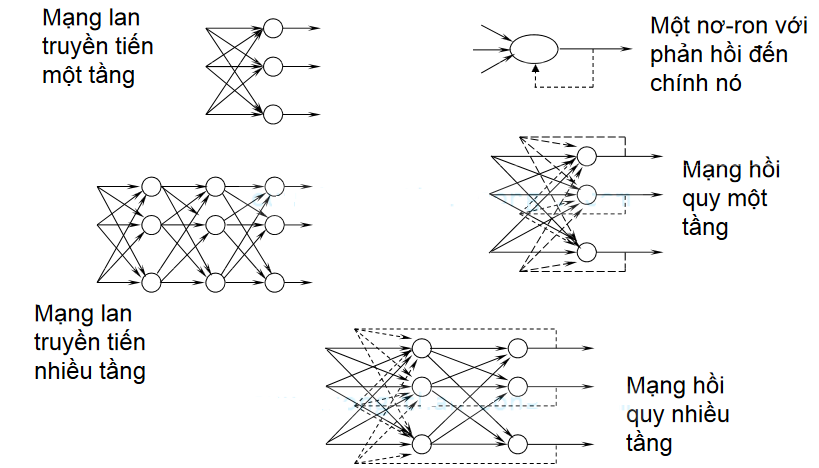
CHÚ Ý:

+ Các Nút [Nơ-ron] trong tầng ẩn (Hidden Neurons | Nodes) KHÔNG tương tác với môi trường ngoài của ANN.

+ ANN “liên kết đầy đủ” [Fully Connected] : mọi Output từ một Tầng liên kết với ALL Nơ-ron tầng kế tiếp.

+ ANN “lan truyền tiến” [Feed-Forward Network]: Output của một nút KHÔNG vào nút cùng tầng hoặc tầng trước; ngược là là ANN “phản hồi” [FeedBack Network]. Nếu cùng tầng là “Phản hồi biên” [Lateral Feedback Network]

+ ANN phản hồi có vòng lặp kín (Closed Loops) là ANN hồi quy [Recurrent Network].



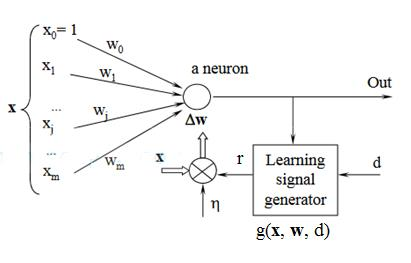
* 1. **Các quy tắc “học” của ANN**

Có 2 kiểu “học” trong ANN

* “Học” bằng “Tham số” (Parameter ML): làm thay đổi thích nghi các “trong số” weight wi của các liên kết trong ANN.
* “Học” thay đổi cấu trúc (Structure ML): làm thay đổi thích nghi cấu trúc ANN = thay đổi số lượng các Nơ-ron và cách liên kết giữa chúng với nhau.

-> 2 cách học trên có thể thực hiện riêng hoặc kết hợp; đa số kiểu 1

* + 1. **Quy tắc “học” bằng trọng số tổng quát**



xi : input của nơ-ron này = output của nơ-ron khác có liên kết với nơ-ron này

Tại bước “học” thứ t, mức độ điều chỉnh vector trọng số **w** tỷ lệ thuận với tích của tín hiệu “học” r(t) với đầu vào **x**(t) :

Δ**w**(t) ≈ r(t) **x**(t)

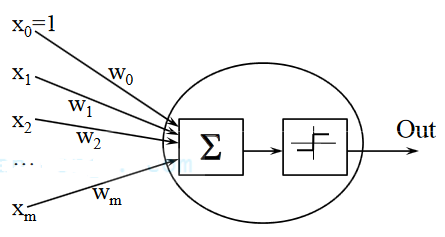
Δ**w**(t) = η r(t) **x**(t) với η > 0 : là “tốc độ học” [Learning Rate]

Với tín hiệu “học” r là một hàm của **w** và **x** với giá trị output mong muốn d : r = g(**w**, **x**, d)

=> Qui tắc “học” bằng “trọng số” tổng quát: Δ**w**(t) = η g(**w**(t) , **x**(t), d(t) ) **x**(t)

* + 1. **Perceptron**

+ Perceptron là một mô hình đơn giản của máy học mô phỏng “nhận thức” nguyên tố (chỉ gồm một Nơ-ron duy nhất).

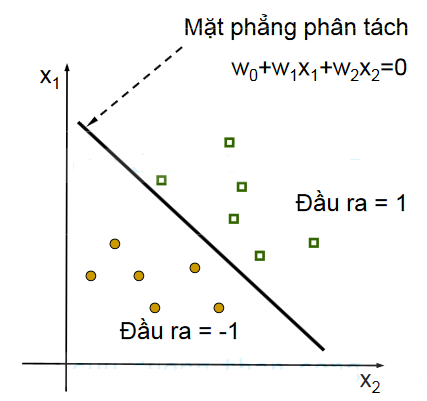


* Sử dụng “hàm tác động” dạng “giới hạn cứng | chặt”:

VD: với vector x thì

**Perceptron minh họa:**

* Có 2 tín hiệu Input: x1 và x2 với trọng số tương ứng w1 và w2 và độ lệch bias là w0

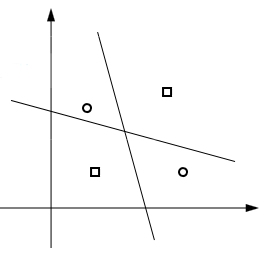


* + 1. **Giải thuật Perceptron**
* Với tập “học” minh họa D = { (x, d) }. Trong đó x là vector input, d = giá trị output mong nuốn: 1 hoặc -1
* Quá trình “học” của perceptron nhằm xác định vector trọng số w giúp perceptron sinh ra giá trị đầu ra d chính xác (-1 or 1) cho mỗi vector “học” x input:

Nếu với vector “học” x input, thì có tình huống xảy ra:

* Được perceptron phân lớp chính xác thì w không thay đổi

Tập mẫu "học" không thể phân tách tuyến tính => giải thuật Perceptron không thể phân lớp chính xác theo cách thông thường của giải thuật perceptron.



* Nếu thực tế d = 1, nhưng perceptron sinh ra output Output = -1 {tức phân lớp sai} thì w cần được thay đổi sao cho giá trị Net(w, x) Tăng lên
* Nếu thực tế d = -1, nhưng perceptron sinh ra output Output = 1 {tức phân lớp sai} thì w cần được thay đổi sao cho giá trị Net(w, x) Giảm xuống.
* **Giải thuật hiệu chỉnh** [nhiều lần = số lượng mẫu “học”] **trực tiếp trên vector trọng số w**

**Perceptron\_Incremental**(D, η)

Khởi\_tạo **w** (wi <- được khởi tạo một giá trị ngẫu nhiên: giá trị nhỏ)

**do**

**for** với mỗi trường hợp “học” (**x**, d) ∈ D

Tính\_Giá\_trị\_output: Out

**if**(Out ≠ d)

**w** ← **w** + η (d – Out) **x**

**end for**

**until** Tất\_cả\_các\_trường\_hợp\_”học”\_trong\_D\_đều cho kết quả phân lớp chính xác

**return w**

----------------biểu diễn tương đương của giải thuật --------------

**Hiệu chỉnh vector trọng số w** [1 lần] **thông qua độ lệch Δw**

**Perceptron\_Batch**(D, η)

Khởi\_tạo **w** (wi <- được khởi tạo một giá trị ngẫu nhiên: giá trị nhỏ)

**do**

Δ**w** ← 0

**for** với\_mỗi\_trường\_hợp\_“học” (**x**, d) ∈ D

Tính\_Giá\_trị\_output: Out

**if**(Out ≠ d)

Δ**w** ← Δ**w** + η (d – Out) **x**

**end for**

**w ← w + Δw**

**until** Tất\_cả\_các\_trường\_hợp\_”học”\_trong\_D\_đều cho kết quả phân lớp chính xác

**return w**

* + 1. **Vấn đề “hội tụ” của giải thuật Perceptron**

+ Giải thuật Perceptron “hội tụ” [Converge] khi:

. Các mẫu “học” có thể phân tách tuyến tính (linearly separable)

. Tốc độ học η đủ nhỏ

+ Nếu tập mẫu “học” không thể phân tách tuyến tính thì giải thuật Perceptron không thể “hội tụ” (tức là, không thể phân lớp chính xác) =khi đó=> phải áp dụng “Quy tắc Delta” [Delta Rule]

. Nhằm giúp giải thuật “hội tụ” về một giá trị xấp xỉ phù hợp nhất của hàm mục tiêu.

. Quy tắc Delta dùng chiến lược Gradient Descent để tìm một vector trọng số w phù hợp nhất với tập học (đang xét D) trong không gian giả thiết các vectors trọng số.

* + 1. **Hàm đánh giá lỗi (Error Function)**

+ Xét ANN có n Nơ-ron output

+ Với mẫu “học” (x, d), giá trị lỗi học (Training Error) gây ra bởi vector trọng số (hiện tại) w:

Lỗi học (Training Error) gây ra bởi vector trọng số (hiện tại) w đối với toàn bộ tập học D

* + 1. **Giải thuật của chiến lược Gradient Descent (giảm độ “dốc”)**

+ Gradient của E (ký hiệu ∇E) là một vector

. Có hướng chỉ đi lên “dốc”

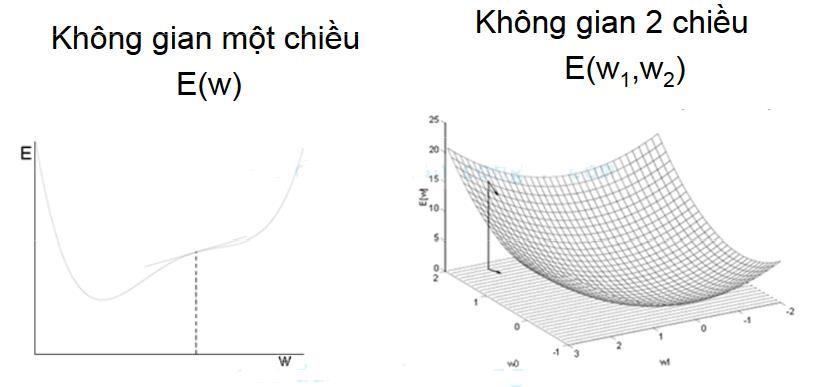
. Có độ dài tỷ lệ thuận với độ “dốc”

+ ∇E xác định hướng gây ra việc TĂNG nhanh nhất giá trị lỗi E (Steepest Increase):

Với N : tổng số trọng số (tổng số lượng liên kết giữa các nơ-ron) trong ANN

+ Vì vậy, hướng gây ra việc GIẢM nhanh nhất (Steepest Decrease)giá trị lỗi E là “phủ định”của ∇E:

-> Yêu cầu (để xác định được giá trị tối ưu): hàm tác động [4.1.3.2] sử dụng trong ANN và đạo hàm của nó phải liên tục đối với các trọng số wi (Xích-Ma OR tiếp tuyến Hyperbolic).



**Giải thuật Gradient Descent:**

**Gradient\_Descent\_Incremental**(D, η)

Khởi\_tạo **w** (wi <- được khởi tạo một giá trị ngẫu nhiên: giá trị nhỏ)

**do**

**for** với\_mỗi\_trường\_hợp\_“học” (**x**, d) ∈ D

Tính\_Giá\_trị\_output: Out

**foreach** giá\_trị\_trọng\_số\_thành\_phần wi

wi ← Δwi - η (δE**x** / δwi)

**end for**

**end for**

**until** thõa\_mãn\_điều\_kiện\_dừng

**return w**

*Với điều\_kiện\_dừng = Số chu kỳ "học" (epochs), ngưỡng "lỗi", . .*

* + 1. **Giải thuật điều chỉnh trọng số adam**

Công thức tổng quát

Với

Với

: (đọc là Ép-si-lon) giá trị cực nhỏ, để tránh trường hợp chia cho 0, thường là 10-8

: (đọc là theta) trọng số được cập nhật lần thứ t

: (đọc là eta) tốc độ học

: (gradient) đạo hàm theo biến theta thứ t

: chỉ số quán tính thứ 1 ()

: chỉ số quán tính thứ 2 ()

: hệ số phụ thuộc (0.9)

: hệ số phụ thuộc (0.999)

Ý tưởng giải thuật: Được hình thành để khác phục hạn chế của 2 thuật toán tối ưu Momentum và Adapvite Learning Rate

* **Momentum**

Sau mỗi lần học, trọng số sẽ được điều chỉnh để đạt được output tối ưu.

Ý tưởng: việc tìm điểm tối ưu như quả cầu đang lau dốc. Với momentum, chúng ta sẽ áp dụng lực quán tính với quả cầu này.

Công thức chung:

Với

Ta được

Với

độ dốc của điểm trước đó (đạo hàm của hàm mất mát với biến là )

: (đọc là gamma) là hằng số quán tính (thường chọn là 0.9)

: xem như lực quán tính của quả cầu.

Như vậy với Momentum, chúng ta nâng cao được khả năng hội tụ của bài toán. Nhưng vẫn còn hạn chế, đó là: khi gần đạt được giá trị tối ưu, thuật toán vẫn khá khó khăn để đạt được vị trí tối ưu này (vì còn lực quán tính ).

* **Adapvite Learning Rate**

Tiếp cận với trường phái khác của việc xử lí bài toán tối ưu, lần này chúng ta sẽ cập nhật lại giá trị để việc hội tụ trở nên nhanh chóng hơn, tôi xin lấy điển hình thuật toán AdaGrad, RMSprop

Với ý tưởng trên, AdaGrad có công thức chung như sau:

Công thức có điểm khác biệc so với thuật toán tối ưu GD, đó chính là được chia cho . (với là hằng số bé để tránh trường hợp chia với 0)

Mỗi lần cập nhật trọng số, sẽ càng nhỏ. (vì sẽ càng lớn với t càng lớn).

Vì thế, sẽ có trường hợp tiến nhanh về 0. Khi đó tốc độ học bị đóng băng.

Để tránh việc này. Thuật toán RMSprop đã hình thành, cụ thể như sau:

sẽ được thay thế bằng

Ví dụ với t = 3 ta tính được như sau,

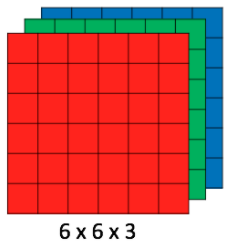
Còn được xác định như sau:

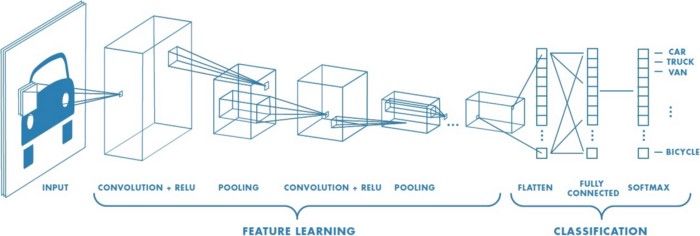
Dễ nhận thấy, < vì thế khả năng bị đóng băng tốc độ học được hạn chế

# 3.4. Giới thiệu về thuật toán Convolution Neural Network

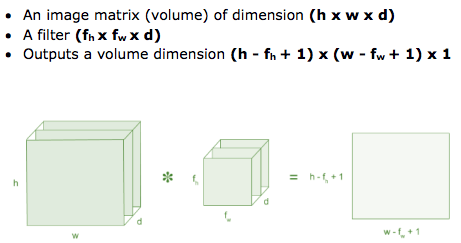
### 3.4.1 Giới thiệu đôi nét

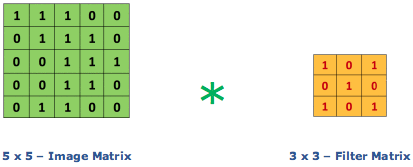
Trong mạng neural, mô hình mạng neural tích chập (CNN) là 1 trong những mô hình để nhận dạng và phân loại hình ảnh. Trong đó, xác định đối tượng và nhận dạng khuôn mặt là 1 trong số những lĩnh vực mà CNN được sử dụng rộng rãi.

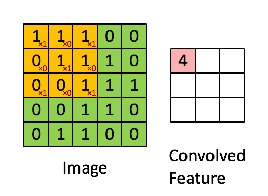
CNN phân loại hình ảnh bằng cách lấy 1 hình ảnh đầu vào, xử lý và phân loại nó theo các hạng mục nhất định (Ví dụ: Đào, Mai, Cúc, ...). Máy tính coi hình ảnh đầu vào là 1 mảng pixel và nó phụ thuộc vào độ phân giải của hình ảnh. Dựa trên độ phân giải hình ảnh, máy tính sẽ thấy H x W x D (H: Chiều cao, W: Chiều rộng, D: Độ dày). Ví dụ: Hình ảnh là mảng ma trận RGB 6x6x3 (3 ở đây là giá trị RGB).

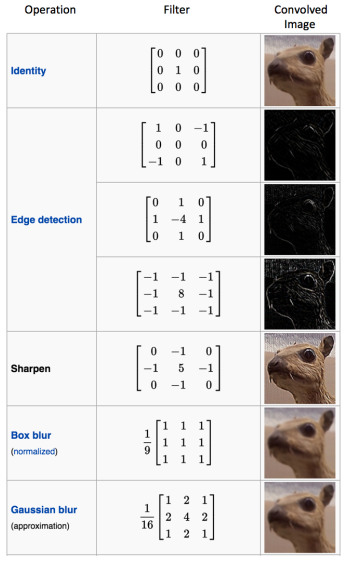
Về kỹ thuật, mô hình CNN để training và kiểm tra, mỗi hình ảnh đầu vào sẽ chuyển nó qua 1 loạt các lớp tích chập với các bộ lọc (Kernals), tổng hợp lại các lớp được kết nối đầy đủ (Full Connected) và áp dụng hàm Softmax để phân loại đối tượng có giá trị xác suất giữa 0 và 1. Hình dưới đây là toàn bộ luồng CNN để xử lý hình ảnh đầu vào và phân loại các đối tượng dựa trên giá trị. 

### 3.4.2. Lớp tích chập - Convolution Layer

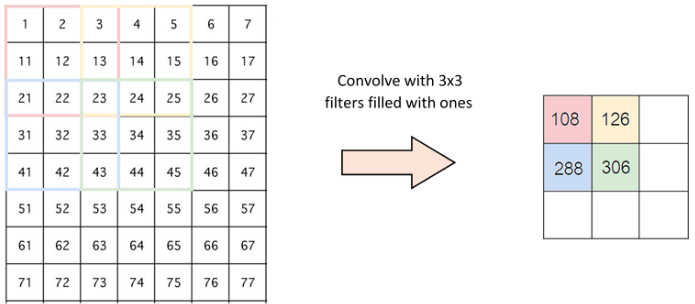
Tích chập là lớp đầu tiên để trích xuất các tính năng từ hình ảnh đầu vào. Tích chập duy trì mối quan hệ giữa các pixel bằng cách tìm hiểu các tính năng hình ảnh bằng cách sử dụng các ô vương nhỏ của dữ liệu đầu vào. Nó là 1 phép toán có 2 đầu vào như ma trận hình ảnh và 1 bộ lọc hoặc hạt nhân.

Xem xét 1 ma trận 5 x 5 có giá trị pixel là 0 và 1. Ma trận bộ lọc 3 x 3 như hình bên dưới.

Sau đó, lớp tích chập của ma trận hình ảnh 5 x 5 nhân với ma trận bộ lọc 3 x 3 gọi là 'Feature Map' như hình bên dưới.

Sự kết hợp của 1 hình ảnh với các bộ lọc khác nhau có thể thực hiện các hoạt động như phát hiện cạnh, làm mờ và làm sắc nét bằng cách áp dụng các bộ lọc. Ví dụ dưới đây cho thấy hình ảnh tích chập khác nhau sau khi áp dụng các Kernel khác nhau.

### 3.4.3. Bước nhảy - Stride

Stride là số pixel thay đổi trên ma trận đầu vào. Khi stride là 1 thì ta di chuyển các kernel 1 pixel. Khi stride là 2 thì ta di chuyển các kernel đi 2 pixel và tiếp tục như vậy. Hình dưới là lớp tích chập hoạt động với stride là 2. 

### 3.4.4. Đường viền - Padding

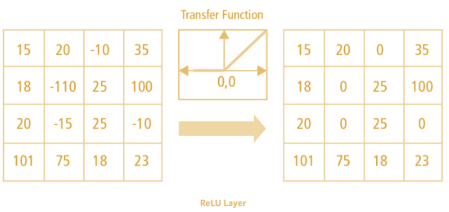
Đôi khi kernel không phù hợp với hình ảnh đầu vào. Ta có 2 lựa chọn:

+Chèn thêm các số 0 vào 4 đường biên của hình ảnh (padding).

+Cắt bớt hình ảnh tại những điểm không phù hợp với kernel.

### 3.4.5. Hàm phi tuyến - ReLU

ReLU viết tắt của Rectified Linear Unit, là 1 hàm phi tuyến. Với đầu ra là: ƒ (x) = max (0, x).

Tại sao ReLU lại quan trọng: ReLU giới thiệu tính phi tuyến trong ConvNet. Vì dữ liệu trong thế giới mà chúng ta tìm hiểu là các giá trị tuyến tính không âm. 

Có 1 số hà phi tuyến khác như tanh, sigmoid cũng có thể được sử dụng thay cho ReLU. Hầu hết người ta thường dùng ReLU vì nó có hiệu suất tốt.

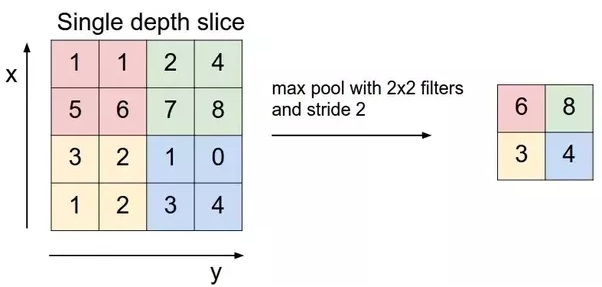
### 3.4.6. Lớp gộp - Pooling Layer

Lớp pooling sẽ giảm bớt số lượng tham số khi hình ảnh quá lớn. Không gian pooling còn được gọi là lấy mẫu con hoặc lấy mẫu xuống làm giảm kích thước của mỗi map nhưng vẫn giữ lại thông tin quan trọng. Các pooling có thể có nhiều loại khác nhau:

+ Max Pooling

+ Average Pooling

+ Sum Pooling

Max pooling lấy phần tử lớn nhất từ ma trận đối tượng, hoặc lấy tổng trung bình. Tổng tất cả các phần tử trong map gọi là sum pooling 

### 3.4.7. Tóm tắt

+ Đầu vào của lớp tích chập là hình ảnh

+ Chọn đối số, áp dụng các bộ lọc với các bước nhảy, padding nếu cần. Thực hiện tích chập cho hình ảnh và áp dụng hàm kích hoạt ReLU cho ma trận hình ảnh.

+ Thực hiện Pooling để giảm kích thước cho hình ảnh.

+ Thêm nhiều lớp tích chập sao cho phù hợp

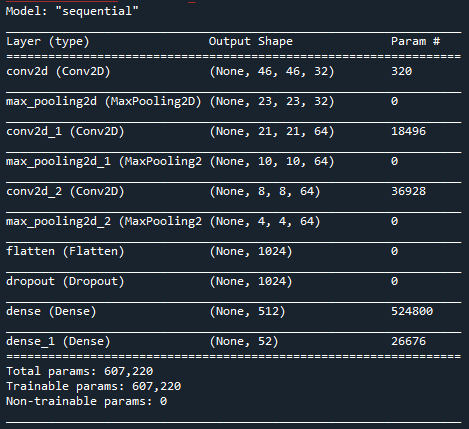
+ Xây dựng đầu ra và dữ liệu đầu vào thành 1 lớp được kết nối đầy đủ (Full Connected)

+ Sử dụng hàm kích hoạt để tìm đối số phù hợp và phân loại hình ảnh.

# GIỚI THIỆU: CÀI ĐẶT VÀ THỰC NGHIỆM HỆ THỐNG ML: CNN

* 1. **Sơ lược về hệ thống**
     1. **Cấu trúc của CNN**

Gồm 1 tầng đầu vào, 8 tầng ẩn và 1 tầng đầu ra



* + 1. **Hàm kích hoạt**

Hàm kích hoạt tại mỗi tầng tương ứng:

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tầng | Hàm kích hoạt | Ghi chú |
| 1 | Conv2D | 'relu' | Tầng Input |
| 2 | MaxPooling2D |  | Tầng dùng lấy các features từ các tập tin đã lưu |
| 3 | Conv2D | 'relu' |  |
| 4 | MaxPooling2D |  | Tầng dùng lấy các features từ các tập tin đã lưu |
| 5 | Flatten |  | Tầng dùng để làm phẳng các layer |
| 6 | Dense | 'relu' |  |
| 7 | Dropout |  |  |
| 8 | Dense | 'relu' |  |
| 9 | Dropout |  |  |
| 10 | Dense | 'softmax' | Tầng Output |

* 1. **Code cài đặt**
     1. Code trích xuất dữ liệu từ tập tin ETL1 tạo file Extention **katakana.npz**

|  |
| --- |
| '''  Trích xuất các tập tin ETL 1 thành file Extention katakana.npz  '''  import struct  from PIL import Image  import numpy as np  def read\_record\_ETL1G(f):      s = f.read(2052)      r = struct.unpack('>H2sH6BI4H4B4x2016s4x', s)      iF = Image.frombytes('F', (64, 63), r[18], 'bit', 4)      iL = iF.convert('P')      return r + (iL,)  def read\_katakana():      katakana = np.zeros([51, 1411, 63, 64], dtype=np.uint8) # 51 characters, 1411 writers, img size = 63\*64      for i in range(7,14):  # Chỉ lấy các tập tin từ ETL1C\_07 đến ETL1C\_13 chứa 48 chữ cái katakana          filename = 'ETL1/ETL1C\_{:02d}'.format(i)          with open(filename, 'rb') as f:              if i!=13: limit = 8              else: limit=3        #Tập tin thứ 13 chỉ bao gồm 3 kí tự              for dataset in range(limit):                  for j in range(1411):                      try :                          r = read\_record\_ETL1G(f)                          katakana[(i - 7) \* 8 + dataset, j] = np.array(r[-1])                      except struct.error:                          pass      np.savez\_compressed("katakana.npz", katakana)  read\_katakana() |

* + 1. Code định dạng kích thước toàn bộ tập ảnh, phân chia thành cái tập học, kiểm tra.

|  |
| --- |
| '''  Định dạng lại toàn bộ tập ảnh vs kích thước 48\*48, trích xuất và phân chia thành các file Extetion  1. katakana\_train\_images.npz         ==> tập ảnh học  2. katakana\_train\_labels.npz         ==> tập label học  3. katakana\_test\_images.npz  4. katakana\_test\_labels.npz  '''  import skimage.transform  import numpy as np  from sklearn.model\_selection import train\_test\_split  kana = np.load("katakana.npz")['arr\_0'].reshape([-1, 63, 64]).astype(np.float32)  kana = kana/np.max(kana) # make the numbers range from 0 to 1  # 51 is the number of different katakana (3 are duplicates so in the end there are 48 classes), 1411 writers.  train\_images = np.zeros([51 \* 1411, 48, 48], dtype=np.float32)  for i in range(51 \* 1411): # change the image size to 48\*48     train\_images[i] = skimage.transform.resize(kana[i], (48, 48))  arr = np.arange(48) # create labels  train\_labels = np.repeat(arr, 1411)  # In the actual code, I combined the duplicate classes here and had 48 classes in the end  # split the images/labels to train and test  train\_images, test\_images, train\_labels, test\_labels = train\_test\_split(train\_images, train\_labels, test\_size=0.2)  np.savez\_compressed("katakana\_train\_images.npz", train\_images)  np.savez\_compressed("katakana\_train\_labels.npz", train\_labels)  np.savez\_compressed("katakana\_test\_images.npz", test\_images)  np.savez\_compressed("katakana\_test\_labels.npz", test\_labels)  '''in ra 25 train\_image đầu tiên đã được trích xuất'''  import matplotlib.pyplot as plt  plt.figure(figsize=(10,10)).patch.set\_facecolor('y')  for i in range(25):      plt.subplot(5,5,i+1)      plt.xticks([])      plt.yticks([])      plt.grid(False)      plt.imshow(train\_images[i], cmap=plt.cm.binary)  plt.show() |

* + 1. Code train model

|  |
| --- |
| import tensorflow as tf  from tensorflow import keras  import numpy as np  from keras.preprocessing.image import ImageDataGenerator  from keras import backend as K  from keras.utils import print\_summary  train\_images = np.load("katakana\_train\_images.npz")['arr\_0']  train\_labels = np.load("katakana\_train\_labels.npz")['arr\_0']  test\_images = np.load("katakana\_test\_images.npz")['arr\_0']  test\_labels = np.load("katakana\_test\_labels.npz")['arr\_0']  if K.image\_data\_format() == "channels\_first": # reshape the image to be able to go through 2D CNN    train\_images = train\_images.reshape(train\_images.shape[0], 1,48,48)    test\_images = test\_images.reshape(test\_images.shape[0], 1,48,48)    shape = (1,48,48)  else:    train\_images = train\_images.reshape(train\_images.shape[0], 48, 48, 1)    test\_images = test\_images.reshape(test\_images.shape[0], 48, 48, 1)    shape = (48,48,1)  datagen = ImageDataGenerator(rotation\_range=15,zoom\_range=0.2)  datagen.fit(train\_images)   # Định nghĩa mô hình Keras: là mô hình tuần tự các lớp ANN `Sequential Model`   # và nạp tất cả các Layers của ANN một lần vào mô hình đó  model = keras.Sequential([        # Định nghĩa mạng CNN được sử dụng với 4 Convolutional layers      # sau là 1 Fully Connected Layer và sử dụng đầu ra là softmax      # Tầng Input      # convolution layer dùng để lấy feature từ image      # Gồm 32 filters, kích thước mỗi filter là (3,3) => tổng số weights = 32\*3\*3 = 288    keras.layers.Conv2D(32, (3,3), activation='relu', input\_shape=shape),    keras.layers.MaxPooling2D(2,2),  # Lấy feature nổi bật và giảm parameter khi training    keras.layers.Conv2D(64, (3,3), activation='relu'),    keras.layers.MaxPooling2D(2,2),    keras.layers.Dropout(0.5),      keras.layers.Conv2D(128, (3,3), activation='relu'),    keras.layers.MaxPooling2D(2,2),    keras.layers.Dropout(0.5),      keras.layers.Conv2D(256, (3,3), activation='relu'),    keras.layers.MaxPooling2D(2,2),    keras.layers.Dropout(0.5),    keras.layers.Flatten(), # Lát phằng layer để fully connection    keras.layers.Dropout(0.5),      #Tầng out put    # Fully connected layers, softmax dùng trong multi classifier    keras.layers.Dense(512, activation='relu'), #Fully conected layers    keras.layers.Dense(52, activation="softmax")  ]) # the model  model.compile(optimizer='adam', loss="sparse\_categorical\_crossentropy", metrics=['accuracy'])  print\_summary(model)  model.fit\_generator(datagen.flow(train\_images,train\_labels,shuffle=True),epochs=30,validation\_data=(test\_images,test\_labels),callbacks = [keras.callbacks.EarlyStopping(patience=8,verbose=1,restore\_best\_weights=True),keras.callbacks.ReduceLROnPlateau(factor=0.5,patience=3,verbose=1)])  model.save("katakana-model.h5") |

* + 1. Code tạo Form giao diện sử dụng tkinter

|  |
| --- |
| from keras.models import load\_model  from tkinter import \*  import tkinter as tk  import win32gui  from PIL import ImageGrab, Image  import numpy as np  import tensorflow as tf  from katakana\_labels import label as lb  model = tf.keras.models.load\_model('katakana-model.h5')  def predict\_digit(img):      #resize image to 48x48 pixels      img = img.resize((48,48))      #convert rgb to grayscale      img = img.convert('L')        img = np.array(img)      #reshaping to support our model input and normalizing      img = img.reshape(1,48,48,1)      img = img/255.0        #predicting the class      res = model.predict([img])[0]      return np.argmax(res), max(res)  class App(tk.Tk):      def \_\_init\_\_(self):          tk.Tk.\_\_init\_\_(self)            self.color\_fg = 'black'          self.old\_x = self.old\_y = 0          self.penwidth=10            # Creating elements          self.canvas = tk.Canvas(self.master, width=300, height=300, bg = "white", cursor="heart")          self.label = tk.Label(self, text="Thinking..", font=("Helvetica", 48))          self.classify\_btn = tk.Button(self, text = "Recognise", command = self.classify\_handwriting)          self.button\_clear = tk.Button(self, text = "Clear", command = self.clear\_all)            # Grid structure          self.canvas.grid(row=0, column=0, pady=2 )          self.label.grid(row=0, column=1,pady=2, padx=2)          self.classify\_btn.grid(row=1, column=1, pady=2, padx=2)          self.button\_clear.grid(row=1, column=0, pady=2)          #self.canvas.bind("<Motion>", self.reset)          self.canvas.bind("<B1-Motion>", self.draw\_lines)          def clear\_all(self):          self.canvas.delete("all")      def classify\_handwriting(self):          handWrite = self.canvas.winfo\_id() # get the handle of the canvas          rect = win32gui.GetWindowRect(handWrite) # get the coordinate of the canvas          img = ImageGrab.grab(rect)          digit, acc = predict\_digit(img)          self.label.configure(text= str(lb[digit])+', '+ str(int(acc\*100))+'%')      def draw\_lines(self, e):          self.old\_x = e.x          self.old\_y = e.y          if self.old\_x and self.old\_y:              self.canvas.create\_line(self.old\_x,self.old\_y,e.x,e.y,width=self.penwidth,fill=self.color\_fg,capstyle=ROUND,smooth=True)  app = App()  app.title('Katakana Recognizor')  mainloop() |

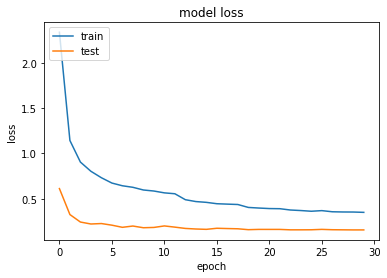
* + 1. Labels

|  |
| --- |
| # Katakana Labels  label = ["ア","イ","ウ","エ","オ",           "カ","キ","ク","ケ","コ",           "サ","シ","ス","セ","ソ",           "タ","チ","ツ","テ","ト",           "ナ","ニ","ヌ","ネ","ノ",           "ハ","ヒ","フ","ヘ","ホ",           "マ","ミ","ム","メ","モ",           "ヤ",    "ユ",    "ヨ",           "ラ","リ","ル","レ","ロ",           "ワ","ヰ","ヱ","ヲ","ン",] |

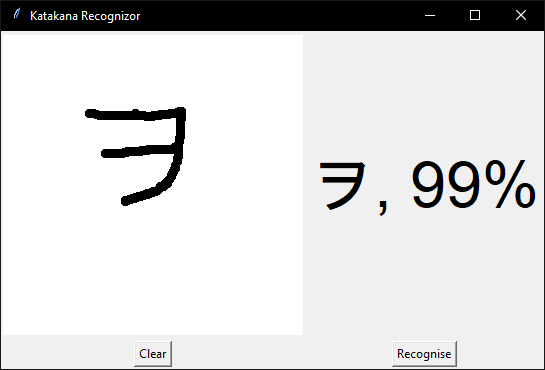
# KẾT QUẢ THỰC NGHIỆM HỆ THỐNG ML: ANN

## **1. Đồ thị**

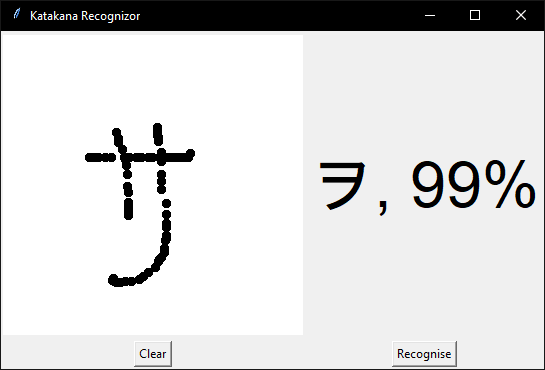
* Đồ thị mô tả độ chính xác:
* Đồ thị mô tả giá trị loss (giá trị lỗi)



## 2. Demo ứng dụng

- Chữ cái ヲ

**- Chữ cái サ**



# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN CỦA ĐỀ TÀI

## **Kết luận**

### Những kết quả đạt được

* Nắm rõ các khái niệm về học máy
* Nắm rõ và biết các cài đặt các thuật toán thường gặp trong học máy
* Biết cách xây dựng một hệ thống học máy
* Cách sử dụng ngôn ngữ Python trong việc cài đặt các thuật toán

### Hạn chế

* Độ chính xác trong việc nhận dạng lúc thực nghiệm chưa cao

## **Hướng phát triển**

### Hướng khắc phục các hạn chế

* Để tăng độ chính xác cho hệ thống, có thể tăng số lượng tập học hoặc tăng số epochs để train mô hình.

### Hướng mở rộng đề tài

Xây dựng thành một ứng dụng có giao diện người dùng, kết hợp với OpenCV để nhận dạng trực tiếp các hình vẽ mà người dùng tương tác với hệ thống qua camera.

# DANH MỤC TÀI LIỆU THAM KHẢO

[1]. Võ Xuân Thể (2020), Bài giảng học phần Học máy, TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM (lưu hành nội bộ).

[2]. Hai, D. M. (2017, 12 19). *[ML] Hồi quy tuyến tính (Linear Regression)*. Được truy lục từ Hai's Blog: https://dominhhai.github.io/vi/2017/12/ml-linear-regression/

# ĐÁNH GIÁ KẾT QUẢ HỌC TẬP HỌC PHẦN

ĐỒ ÁN HP THAY CHO BÀI THI KẾT THÚC HỌC PHẦN

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Hình thức KT** | **Nội dung** | **Chuẩn đầu ra đánh giá** | **Trình độ năng lực** | **Phương pháp đánh giá** | **Công cụ đánh giá** | **Tỉ lệ (%)** |
| **BÀI TẬP LẬP TRÌNH** | | | | | |  |
| BL#1 | Bài tập lập trình theo từng Buổi thực hành = tính bình quân | G1.1 | 3 | Bài tập | Bài tập | 40 |
| G2.1 | 3 |
| G3.1  G3.2 | 4  3 |
| G4.1 | 5 |
| **ĐỒ ÁN HỌC PHẦN (Project)** | | | | | |  |
| ĐA#1 | Phát triển một hệ thống học máy đơn giản:  + Đề tài chung theo nhóm: Giải thuật ML  + Xây dựng và thực nghiệm giải thuật theo bài toán cá nhân  => báo cáo vào tuần cuối. | G1.1 | 3 | Báo cáo, Demo và thuyết trình | Rubric | 40 |
| G2.1 | 3 |
| G3.1 | 4 |
| G3.2  G4.1 | 3  5 |
| G2.1 | 4 |
| G3.1 | 4 |
| G3.2  G4.1 | 3  5 |
| **BÀI KIỂM TRA LÝ THUYẾT** | | | | | |  |
| BT#1 | Bài kiểm tra lý thuyết: Hiểu các giải thuật  học máy phổ biến?**CĐR 2.2** | G1.1 | 3 | Kiểm tra | Câu hỏi | 20 |
| G2.1 | 3 |
| G3.2 | 3 |

1. Đây là khái niệm của tác giải bài báo cáo này. [↑](#footnote-ref-1)