TRƯỜNG ĐẠI HỌC

SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH

HCMC University of Technology and Education



KHOA ĐÀO TẠO CHẤT LƯỢNG CAO

NGÀNH CÔNG NGHỆ THÔNG TIN



BÁO CÁO ĐỒ ÁN I

TENSORGRAM

PHẦN MỀM TẠO DIAGRAM

CHO MÔ HÌNH TENSORFLOW

|  |  |
| --- | --- |
| Nhóm sinh viên thực hiện: | |
| Huỳnh Quốc Hoàng Vương | 17110256 |
| Bùi Minh Trung  Phạm Quốc Việt | 17110243  17110254 |

GVHD: TS. Huỳnh Xuân Phụng

Tp. Hồ Chí Minh, tháng 11 – 2019

ĐIỂM SỐ

|  |  |  |  |
| --- | --- | --- | --- |
| TIÊU CHÍ | NỘI DUNG | TRÌNH BÀY | TỔNG |
| ĐIỂM |  |  |  |

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

Giáo viên hướng dẫn

(*ký và ghi họ tên*)

Huỳnh Xuân Phụng

LỜI CẢM ƠN

Để hoàn thành tốt đề tài và bài báo cáo này, chúng em xin gửi lời cảm ơn chân thành đến giảng viên, tiến sĩ Huỳnh Xuân Phụng, người đã trực tiếp hỗ trợ chúng em trong suốt quá trình làm đề tài. Chúng em cảm thầy đã đưa ra những lời khuyên từ kinh nghiệm thực tiễn của mình để định hướng cho chúng em đi đúng với yêu cầu của đề tài đã chọn, luôn giải đáp thắc mắc và đưa ra những góp ý, chỉnh sửa kịp thời giúp chúng em khắc phục nhược điểm và hoàn thành tốt cũng như đúng thời hạn Khoa đã đề ra.

Chúng em cũng xin gửi lời cảm ơn chân thành các quý thầy cô trong khoa Đào tạo Chất Lượng Cao nói chung và ngành Công Nghệ Thông Tin nói riêng đã tận tình truyền đạt những kiến thức cần thiết giúp chúng em có nền tảng để làm nên đề tài này, đã tạo điều kiện để chúng em có thể tìm hiểu và thực hiện tốt đề tài. Cùng với đó, chúng em xin được gửi cảm ơn đến các bạn cùng khóa đã cung cấp nhiều thông tin và kiến thức hữu ích giúp chúng em có thể hoàn thiện hơn đề tài của mình.

Đề tài và bài báo cáo được chúng em thực hiện trong khoảng thời gian ngắn, với những kiến thức còn hạn chế cùng nhiều hạn chế khác về mặt kĩ thuật và kinh nghiệm trong việc thực hiện một dự án phần mềm. Do đó, trong quá trình làm nên đề tài có những thiếu sót là điều không thể tránh khỏi nên chúng em rất mong nhận được những ý kiến đóng góp quý báu của các quý thầy cô để kiến thức của chúng em được hoàn thiện hơn và chúng em có thể làm tốt hơn nữa trong những lần sau. Chúng em xin chân thành cảm ơn.

Cuối lời, chúng em kính chúc quý thầy, quý cô luôn dồi dào sức khỏe và thành công hơn nữa trong sự nghiệp trồng người. Một lần nữa chúng em xin chân thành cảm ơn.

TP.HCM, ngày 10 tháng 11 năm 2019

Nhóm sinh viên thực hiện

MỤC LỤC

[Danh mục các hình 1](#_Toc24298305)

[Danh mục các bảng 2](#_Toc24298306)

[Chương 1: Tổng quan chương trình 4](#_Toc24298307)

[1. Giới thiệu chung 4](#_Toc24298308)

[1.1. Về đồ án phần mềm vẽ Diagram TensorFlow 4](#_Toc24298309)

[1.1.1. Yêu cầu đồ án 4](#_Toc24298310)

[1.1.2. Phân tích đồ án 4](#_Toc24298311)

[1.1.3. Phương hướng thực hiện 4](#_Toc24298312)

[1.2. Machine Learning, Tensorflow và Layers API 4](#_Toc24298313)

[1.2.1. Lí thuyết Machine Learning cơ bản 4](#_Toc24298314)

[1.2.2. Thư viện ML TensorFlow 5](#_Toc24298315)

[1.2.3. Layers API của TensorFlow 5](#_Toc24298316)

[1.2.3.1. Artificial Neural Network (ANN) 5](#_Toc24298317)

[1.2.3.2. Layers API 6](#_Toc24298318)

[2. Đặc tả phần mềm TensorGram 6](#_Toc24298319)

[2.1. Phần mềm TensorGram 6](#_Toc24298320)

[2.1.1. Giới thiệu về phần mềm TensorGram 6](#_Toc24298321)

[2.1.2. Use Case Diagram 7](#_Toc24298322)

[2.1.3. Dữ liệu đầu vào – đầu ra (input - output) 7](#_Toc24298323)

[2.1.3.1. Phân tích dữ liệu đầu vào (input) 7](#_Toc24298324)

[2.1.3.2. Phân tích dữ liệu đầu ra (output) 8](#_Toc24298325)

[2.1.4. Tính năng chính 8](#_Toc24298326)

[2.1.5. Ứng dụng 8](#_Toc24298327)

[2.2. Yêu cầu kĩ thuật 8](#_Toc24298328)

[2.3. Công cụ và công nghệ sử dụng 9](#_Toc24298329)

[Chương 2: Kế hoạch thực hiện 9](#_Toc24298330)

[1. Kế hoạch 9](#_Toc24298331)

[2. Phân công công việc 10](#_Toc24298332)

[Chương 3: Thiết kế phần mềm 11](#_Toc24298333)

[1. Thuật toán 11](#_Toc24298334)

[1.1. Đọc input 11](#_Toc24298335)

[1.2. Dựng hình 11](#_Toc24298336)

[2. Thiết kế giao diện 13](#_Toc24298337)

[2.1. Giao diện chương trình 13](#_Toc24298338)

[2.2. Đặc tả giao diện 13](#_Toc24298339)

[3. Thiết kế lớp 17](#_Toc24298340)

[3.1. Thiết kế lớp cho các Layer của TensorFlow Layers API 17](#_Toc24298341)

[3.1.1. Tổng quan 17](#_Toc24298342)

[3.1.2. Chi tiết 20](#_Toc24298343)

[3.1.3. Đặc tả lớp 21](#_Toc24298344)

[3.1.4. Đặc tả các phương thức trong lớp 22](#_Toc24298345)

[3.2. Thiết kế lớp chức năng 23](#_Toc24298347)

[3.2.1. Tổng quan 23](#_Toc24298348)

[3.2.2. Đặc tả lớp 24](#_Toc24298349)

[3.2.3. Đặc tả các phương thức trong lớp 24](#_Toc24298350)

[Chương 4: Cài đặt và kiểm thử 29](#_Toc24298351)

[Chương 5: Kết luận và hướng phát triển 33](#_Toc24298352)

[1. Kết luận 33](#_Toc24298353)

[2. Hướng phát triển 34](#_Toc24298354)

[Tài liệu tham khảo 34](#_Toc24298355)

# Danh mục các hình

[Hình 1: Mô hình mạng Neural nhân tạo 5](#_Toc24298356)

[Hình 2: Use case diagram 7](#_Toc24298357)

[Hình 3: Minh hoạ dữ liệu đầu vào. 7](#_Toc24298358)

[Hình 4: Minh hoạ dữ liệu trả ra. 8](#_Toc24298359)

[Hình 5: Cấu trúc file script mô tả một layer trong Model Tensorflow ANN 11](#_Toc24298360)

[Hình 6: Giao diện phần mềm 13](#_Toc24298361)

[Hình 7: UML Diagram biểu diễn các lớp trong phần mềm. 20](#_Toc24298362)

[Hình 8: Kiểm thử 1 29](#_Toc24298363)

[Hình 9: Kiểm thử 2 30](#_Toc24298364)

[Hình 10: Kiểm thử 3 31](#_Toc24298365)

[Hình 11: Kiểm thử 4 31](#_Toc24298366)

[Hình 12: Kiểm thử 5 32](#_Toc24298367)

# Danh mục các bảng

[Bảng 1: Kế hoạch theo tuần 9](#_Toc24298368)

[Bảng 2: Phân công công việc & đóp góp của mỗi sinh viên 10](#_Toc24298369)

[Bảng 3: Đặc tả giao diện 13](#_Toc24298370)

[Bảng 4: Các Class và Attributes ứng với các layer. 17](#_Toc24298371)

[Bảng 5: Chi tiết chức năng các layer trong ANN Model 19](#_Toc24298372)

[Bảng 6: Danh mục các lớp cho các Layer của TensorFlow Layers API 21](#_Toc24298373)

[Bảng 7: Đặc tả các phươn thức trong lớp Layer 22](#_Toc24298374)

[Bảng 8: Đặc tả các lớp chức năng 24](#_Toc24298375)

[Bảng 9: Đặc tả các phương thức trong lớp Render\_MasterControl 24](#_Toc24298376)

[Bảng 10: Đặc tả các phương thức trong lớp TensorModel 25](#_Toc24298377)

[Bảng 11: Đặc tả các phương thức trong lớp ConnectorRender\_Control 26](#_Toc24298378)

[Bảng 12: Đặc tả các phương thức trong lớp SlidePanel\_Control 26](#_Toc24298379)

[Bảng 13: Đặc tả các phương thức trong lớp Arrow 27](#_Toc24298380)

[Bảng 14: Đặc tả các phương thức trong lớp TextInput\_Hander 27](#_Toc24298381)

# Chương 1: Tổng quan chương trình

## Giới thiệu chung

### Về đồ án phần mềm vẽ Diagram TensorFlow

#### Yêu cầu đồ án

Thiết kế và xây dựng phần mềm hướng đối tượng giải quyết yêu cầu vẽ Diagram sử dụng TensorFlow Layer API cho mô hình mạng thần kinh nhân tạo từ mô tả bằng Python scripts của người dùng.

#### Phân tích đồ án

* Xây dựng phần mềm hướng đối tượng.
* Phân tích text (Đọc string) để lấy dữ liệu đầu vào.
* Dữ liệu đầu ra đưa tới cho người dùng dạng đồ hoạ.

#### Phương hướng thực hiện

* Xây dựng phần mềm hướng đối tượng bằng C#, đáp ứng cả 4 tính chất: Kế thừa, đóng gói, đa hình và trù tượng.
* Ứng dụng công nghệ Windows Presentation Foundation (WPF) vào thiết kế giao diện người dùng.

### Machine Learning, Tensorflow và Layers API

#### Lí thuyết Machine Learning cơ bản

Những năm gần đây, Artificial Intelligence (Trí Tuệ Nhân Tạo - AI), và cụ thể hơn là Machine Learning (Học Máy hoặc Máy Học) nổi lên như một bằng chứng của cuộc cách mạng công nghiệp lần thứ tư. AIđang len lỏi vào mọi lĩnh vực trong đời sống mà có thể chúng ta không nhận ra. Xe tự hành của Google và Tesla, hệ thống tự tag khuôn mặt trong ảnh của Facebook, trợ lý ảo Siri của Apple, trợ lí ảo Alexa của Amazon, Cortana của Microsoft, hệ thống gợi ý sản phẩm của Amazon, hệ thống gợi ý phim của Netflix, máy chơi cờ vây AlphaGo của Google DeepMind, …, chỉ là một vài trong vô vàn những ứng dụng của AI.

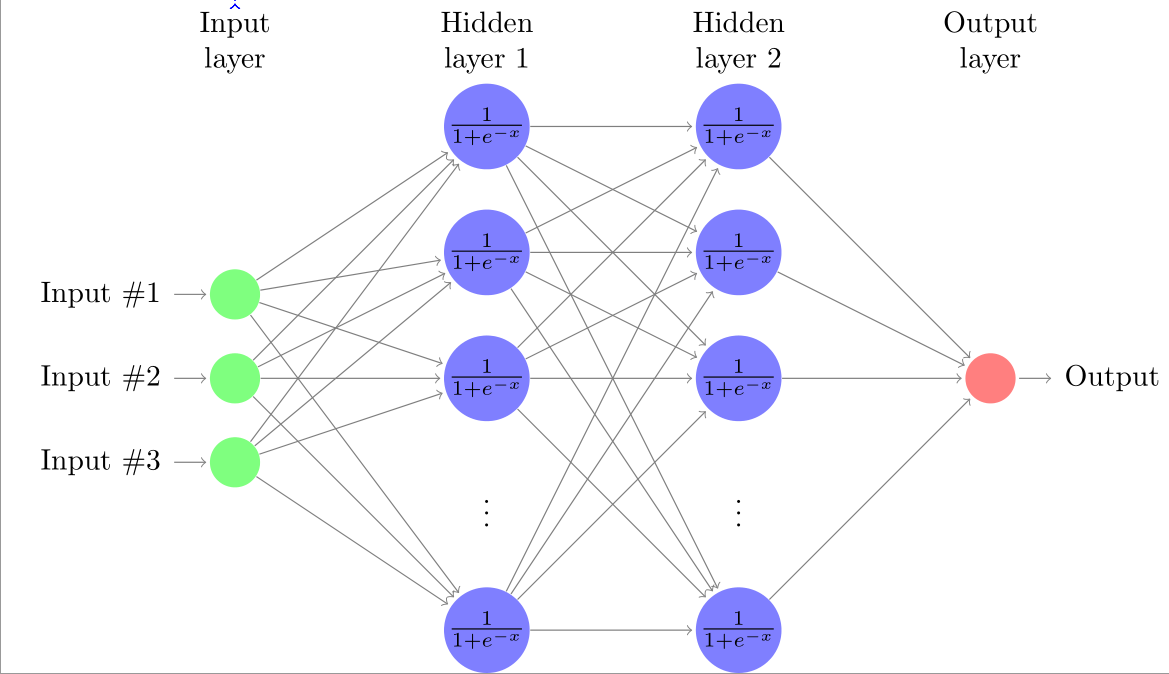
Machine Learning là một tập con của AI. Theo định nghĩa của Wikipedia, Machine learning is the subfield of computer science that “gives computers the ability to learn without being explicitly programmed”. Nói đơn giản, Machine Learning là một lĩnh vực nhỏ của Khoa Học Máy Tính, nó có khả năng tự học hỏi dựa trên dữ liệu đưa vào mà không cần phải được lập trình cụ thể.

#### Thư viện ML TensorFlow

TensorFlow là một thư viện Machine Learining được Google phát triển và phát hành vào tháng 10 năm 2015. Thư viện này hỗ trợ xây dựng các mô hình Machine Learning rất phức tạp qua những API cực kì ngắn gọn. Các mô hình Machine Learning phát triển trên TensorFlow có thể được sử dụng trên nhiều nền tảng khác nhau (từ Smartphone tới Distributed Servers) và trên cả CPUs lẫn GPUs.

#### Layers API của TensorFlow

##### Artificial Neural Network (ANN)



Hình 1: Mô hình mạng Neural nhân tạo

Mạng Neural nhân tạo là sự kết hợp của của các tầng perceptron hay còn được gọi là perceptron đa tầng.

Một mạng ANN sẽ có 3 kiểu tầng:

* Tầng vào (Input layer): Là tầng bên trái cùng của mạng thể hiện cho các đầu vào của mạng.
* Tầng ra (Output layer): Là tầng bên phải cùng của mạng thể hiện cho các đầu ra của mạng.
* Tầng ẩn (Hidden layer): Là tầng nằm giữa tầng vào và tầng ra thể hiện cho việc suy luận logic của mạng).

Một ANN chỉ có 1 tầng vào và 1 tầng ra nhưng có thể có nhiều tầng ẩn. Trong mạng ANN, mỗi nút mạng là một node đơn lẻ nhưng chức năng của chúng có thể khác nhau. Tuy nhiên trong thực tế người ta thường để chúng cùng dạng với nhau để tính toán cho thuận lợi. Ở mỗi tầng, số lượng các nút mạng có thể khác nhau tuỳ thuộc vào bài toán và cách giải quyết.

##### Layers API

Layers API là một module của TensorFlow, được tạo ra bởi François Chollet, tác giả của bộ thư viện Keras với chức năng tương tự. Layers API được dùng để tạo ra một ANN.

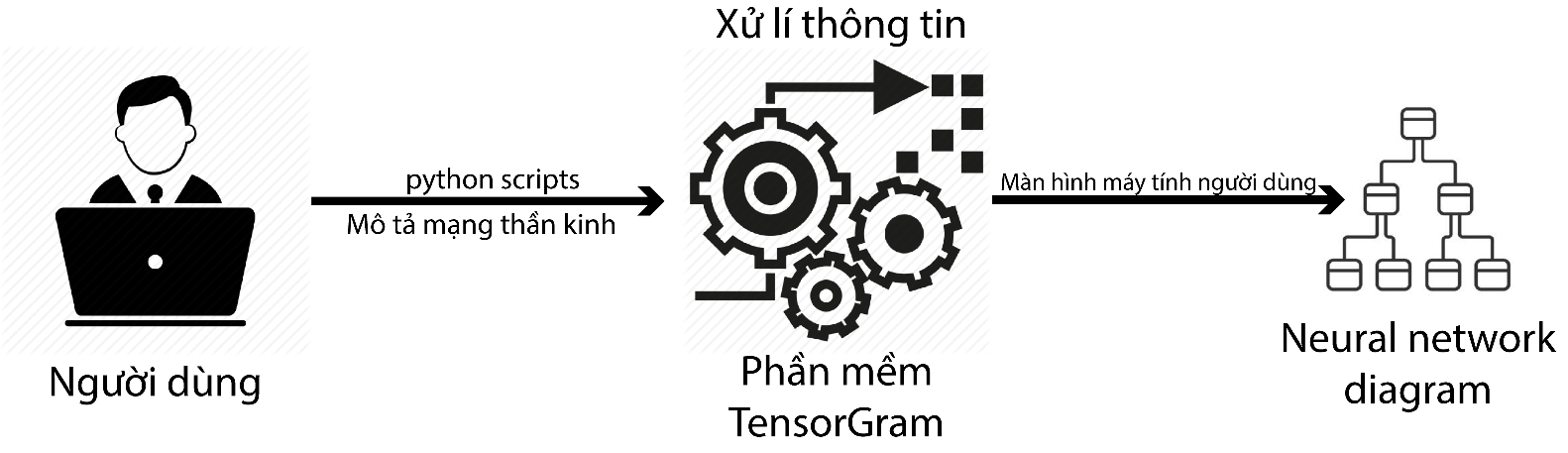
## Đặc tả phần mềm TensorGram

### Phần mềm TensorGram

#### Giới thiệu về phần mềm TensorGram

TensorGram là một phần mềm nhỏ gọn (portable) chạy mà không cần cài đặt dùng để đồ thị hoá một *ANN* được định nghĩa bởi người dùng thông qua đoạn Python Scripts nhập trực tiếp vào khung soạn thảo trong chương trìnhthành dạng TensorFlow Layers và hiển thị lên cho người dùng.

#### Use Case Diagram

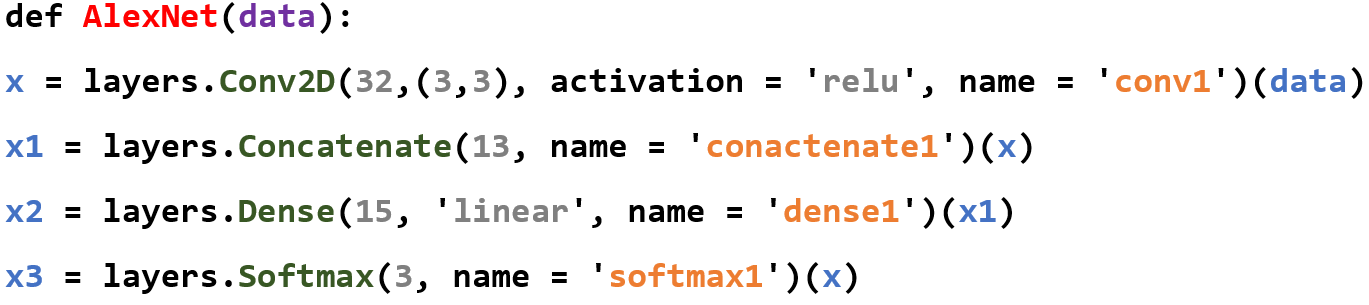


Hình 2: Use case diagram

#### Dữ liệu đầu vào – đầu ra (input - output)

* + Input: Python Scripts chứa mô tả về một mạng thần kinh nhân tạo.
  + Output: Hiển thị lên màn hình máy tính người dùng một Diagram dưới dạng TensorFlow Layer API của mạng thần kinh nhân tạo được mô tả ở input.

##### Phân tích dữ liệu đầu vào (input)

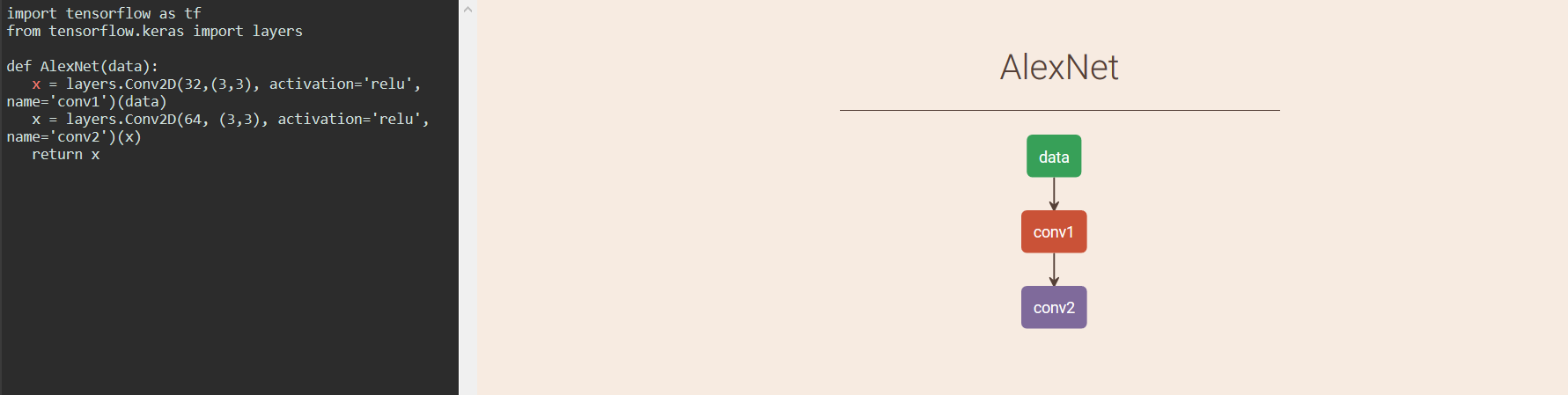


Hình 3: Minh hoạ dữ liệu đầu vào.

Đoạn scripts đầu vào có thể chia làm 3 phần:

* + Dòng đầu tiên định nghĩa Model và InputLayer
  + Các dòng tiếp theo mô tả các Layer trong Model và các kết nối giữa chúng
  + Dòng cuối cùng là output của cả Model

##### Phân tích dữ liệu đầu ra (output)



Hình 4: Minh hoạ dữ liệu trả ra.

#### Tính năng chính

* Tạo Diagram về ANN dưới dạng TensorFlow Layer và hiển thị trên người dùng.
* Thể hiện thông tin chi tiết về từng Layer trong mạng Layer đó.
* Tìm kiếm Layer theo tên.
* Thực hiện các tương tác trên vùng hiển thị diagram:
  + Phóng to diagram.
  + Thu nhỏ diagram .
  + Di chuyển khung nhìn để tập trung đến các vùng khác nhau của diagram.

#### Ứng dụng

Giúp người dùng có cái nhìn trực quan về ANN của mình dưới dạng TensorFlow Layer Diagram mà không cần phải cài đặt và sử dụng TensorBoard cồng kềnh và phức tạp, cũng như thay vì phải xây dựng lại đầy đủ một ANN thì giờ đây người dùng chỉ cần vài câu python scripts là có thể xây dựng cho mình một ANN On-The-Go (Tất nhiên là model này vẫn chưa có dữ liệu).

### Yêu cầu kĩ thuật

* Thực hiện được yêu cầu mà đồ án đề ra.
* Áp dụng lập trình hướng đối tượng và các công nghệ phần mềm mới.
* Dung lượng phần mềm nhẹ, chạy ổn định.

### Công cụ và công nghệ sử dụng

* Xây dựng phần mềm bằng công nghệ WPF trên nền .NET Framework 4.7.2
* Thiết kế giao diện người dùng (GUI) bằng Blend for Visual Studio 2019 (Code Xaml)
* Thiết kế View Model và Data Model bằng Visual Studio 2019 (Code C#)

# Chương 2: Kế hoạch thực hiện

## Kế hoạch

Bảng 1: Kế hoạch theo tuần

|  |  |
| --- | --- |
| Tuần | Công việc |
| 5 | Tìm hiểu về Machine Learning và mô hình Mạng thần kinh nhân tạo |
| 6 | Tìm hiểu về Tensorflow, bộ Layer API và công nghệ Windows Presentation Foundation |
| 7 | Phân tích input, xây dựng và cài đặt thuật toán đọc input. Bắt đầu thiết kế giao diện |
| 8 | Xây dựng và cài đặt các lớp cho TensorFlow Layer API và các lớp chức năng của phần mềm |
| 9 | Xây dựng và cài đặt thuật toán dựng hình |
| 10 | Xây dựng, cài đặt chức năng tìm kiếm, hoàn thành phần mềm. Soát lỗi, kiểm thử phần mềm. |
| 11 | Viết báo cáo và làm file trình chiếu phục vụ việc báo cáo và thuyết trình đồ án |

## Phân công công việc

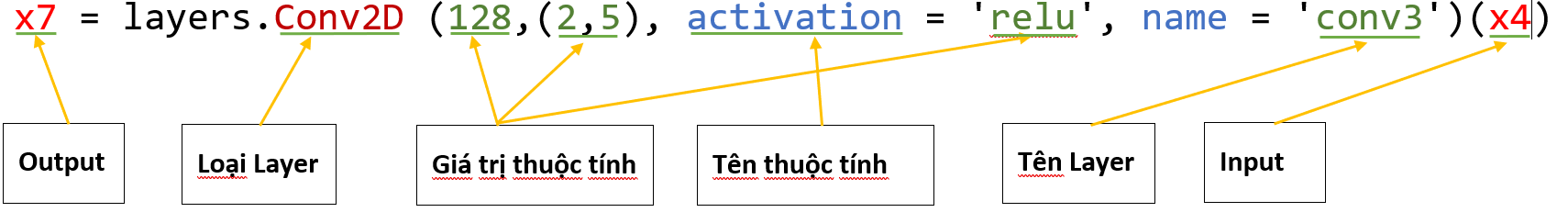
Bảng 2: Phân công công việc & đóp góp của mỗi sinh viên

|  |  |  |  |
| --- | --- | --- | --- |
| TT | Tên sinh viên | Miêu tả công việc | Đóng góp |
| 1 | Huỳnh Q.H. Vương | * Thiết kế chính các lớp của TensorFlow Layer API * Thiết kế chính các lớp chức năng cho phần mềm * Thiết kế thuật toán đọc input và dựng đồ hoạ | 60% |
| 2 | Bùi Minh Trung | * Thiết kế giao diện phần mềm và báo cáo | 20% |
| 3 | Phạm Quốc Việt | * Báo cáo và cài đặt và kiểm thử. | 20% |

# Chương 3: Thiết kế phần mềm

## Thuật toán

### Đọc input



Hình 5: Cấu trúc file script mô tả một layer trong Model Tensorflow ANN

Như hình trên đã biểu diễn, một câu scripts miêu tả một Layer đều có thuộc tính bao gồm:

* + Output.
  + Loại Layer.
  + Tên Layer.
  + Input.

Vì lí do đó, các câu script sẽ được xử lí thô thông qua lớp InputHander. Tại đây, các câu script sẽ được:

* + Chuẩn hoá (Bỏ các khoảng trắng thừa)
  + Tách từ (trim) theo các dấu hiệu (dấu = . , ( ) ) tư đó tách ra 4 thuộc tính chung của layer mà câu scripts đó miêu tả.
  + Sau khi xác định loại layer ở bước trên, chương trình tạo layer dựa trên loại layer và đồng thời truyền ba thông số vừa đọc được từ script.
  + Cuối cùng, dữ liệu thô về các thuộc tính riêng của từng layer được trích ra từ script và truyền vào cho hàm ReadAttribute trong class ứng với Layer nó hiện thực để xử lí. Đối với một số Layer đặc biệt (Add, Average, InputLayer) không có thuộc tính riêng, không cần thực hiện bước này.

### Dựng hình

Để dựng được diagram theo mô tả người dùng dưới dạng đồ hoạ, cần trải qua các quá trình :

* Xác định lớp cha – lớp con.
* Xây dựng cây con phân lớp bằng Dictionary với key là Level (biểu diễn từ số nguyên từ 1 🡪 n) và giá trị là một danh sách các Layer thuộc Level đó. Về cách phân chia level thì InputLayer sẽ ở level 1 và OutputLayer sẽ ở level cuối cùng, Layer con ở dưới Layer cha tối thiểu 1 level.
* Dựng đồ hoạ cho diagram dựa theo cây con phân lớp vừa tạo ở trên.
* Dựng đồ hoạ cho các kết nối giữa các Layer.

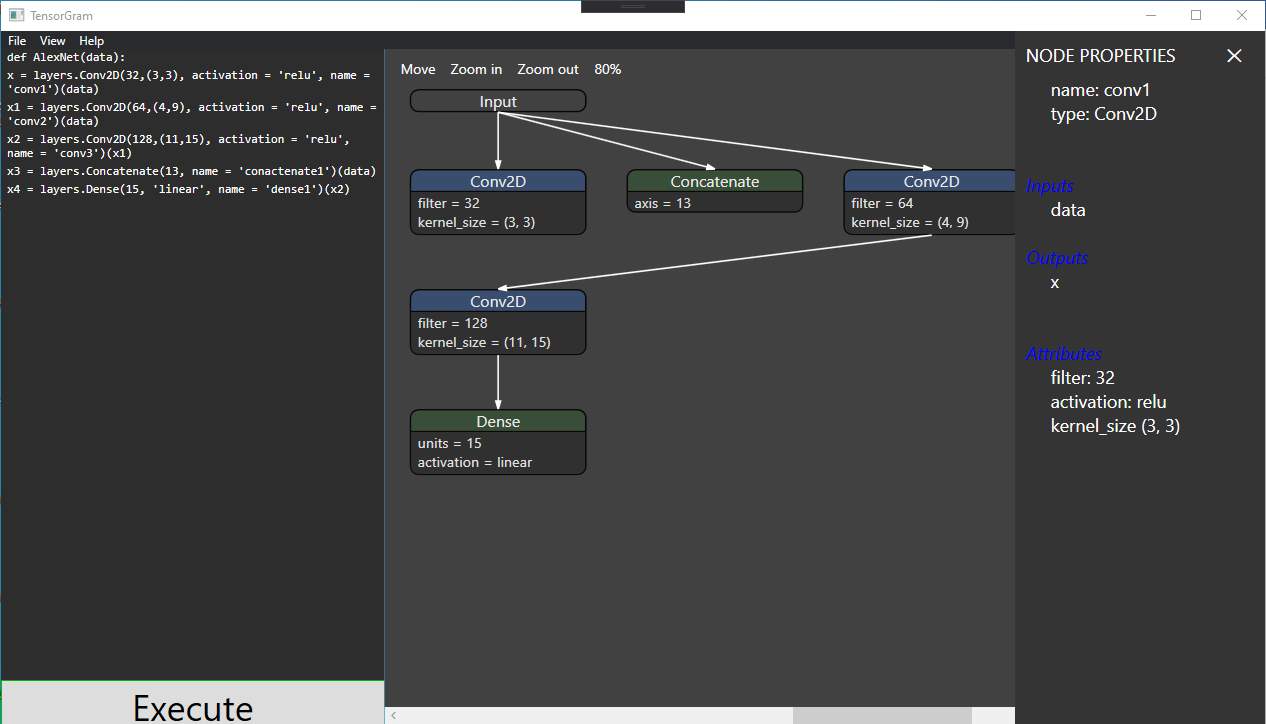
## Thiết kế giao diện

### Giao diện chương trình

Chức năng di chuyển và thu phóng trên khu vực hiển thị

Nơi hiển thị TensorFlow Layer Diagram

Thanh công cụ



Hình : Giao diện phần mềm

Khu vực hiển thị thông tin của Layer được chọn / Chức năng tìm kiếm

Nơi nhập python scripts chứa mô tả về ANN

Nút thực thi python scripts 🡪 Diagram

### Đặc tả giao diện

Bảng 3: Đặc tả giao diện

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TT | Tên | Phân loại | Make-up | Chức năng - chú thích |
| 1 | HeaderMenu | Menu | <Menu Grid.ColumnSpan="2" Height="18" VerticalAlignment="Top" Background="#FF292A2D" Foreground="White">  <MenuItem Header="\_File" Foreground="White">  <MenuItem Header="\_Open" Foreground="Black" />  <MenuItem Header="\_Export" Foreground="Black" />  <Separator />  <MenuItem Header="\_Exit" Foreground="Black" Click="MenuItem\_Click" />  </MenuItem>  <MenuItem Header="View" Foreground="White">  <MenuItem Header="\_Find" Foreground="Black" Click="MenuItem\_Find\_Click" />  </MenuItem>  <MenuItem Header="Help" Foreground="White">  </MenuItem>  </Menu> | Menu chức năng cho phần mềm  Cấu trúc menu:   * File * Open * Export * Exit * View * Find * Help |
| 2 | txtCodeInput | RichTextBox | <RichTextBox x:Name="txtCodeInput" Background="#FF2D2D2D" Foreground="White" FontFamily="Consolas" Block.LineHeight="4" Height="631" BorderBrush="#FF2D2D2D" SelectionBrush="{x:Null}">  <FlowDocument>  <Paragraph>  <Run Text=""/>  </Paragraph>  </FlowDocument>  </RichTextBox> | Nơi người dùng nhập input cho chương trình |
| 3 | bntExec | Button | <Button x:Name="bntExec" Padding="0, 0, 0, 0" Height="53" BorderBrush="{x:Null}" Content="Execute" FontSize="36" Click="BntExec\_Click" HorizontalContentAlignment="Center" VerticalContentAlignment="Center"/> | Dùng để báo hiệu cho chương trình bắt đầu xử lí và xuất kết quả cho dữ liệu người dùng vừa nhập. |
| 2 | HolderFor  UserInput | StackPanel | <StackPanel x:Name="HolderForUserInput" Grid.Column="0" Background="#FF05B405" Margin="0,18,0,0"/> | Chứa nhóm hai control bntExec và txtCodeInput. |
| 3 | MainCanvas | Canvas | <Canvas x:Name="MainCanvas" Background="#FF414141" Height="5000" HorizontalAlignment="Center" Width="5000" Margin="-2500,0,-2500,-4300" VerticalAlignment="Top"/> | Nôi hiển thị digram sau khi xử lí input, kích thước 5000x5000 px. |
| 4 | MainScroll-  Viewer | ScrollViewer | <ScrollViewer x:Name="MainScrollViewer" Grid.Column="1" VerticalScrollBarVisibility="Visible" HorizontalScrollBarVisibility="Visible" Background="{DynamicResource {x:Static SystemColors.HighlightBrushKey}}" Margin="0,18,0,0"> | Tạo khung nhìn có kích thước 886x683 px cho MainCanvas. Khung nhìn này có thể dịch chuyển được trong vùng Canvas bằng hai thanh cuộn dọc và ngang. |
| 5 | SlideMenu\_  StackPanel | StackPanel | <StackPanel Grid.Column="1" Panel.ZIndex="2" Name="SlideMenu\_StackPanel" Orientation="Horizontal" Width="250" HorizontalAlignment="Right" Margin="0,18,-250,0"> | Tạo một khung năm bên phải cửa sổ chính của chương trình, có thể ẩn và hiện theo ý người dùng, điều khiển bởi hai Storyboard là ShowMenu và HideMenu. Control này là nơi hiển thị thông tin chi tiết của Layer mà người dùng chỉ định, đồng thời là nơi người dùng thực hiện chức năng tìm kiếm Layer. |
| 6 | SlidePanel\_  Title | Label | <Label x:Name="SlidePanel\_Title" Content="NODE PROPERTIES" HorizontalAlignment="Left" Margin="4.706,5.736,0,0" VerticalAlignment="Top" FontSize="18" Foreground="White"/> | Thuộc SlideMenuStackPanel, hiện thị chức năng hiện tại của SlideMenu (Hiển thị thông tin hay tìm kiếm). |
| 7 | SlidePanel\_  TextBlock | TextBlock | <TextBlock x:Name = "SlidePanel\_TextBlock" Margin="10,45,10,10" TextWrapping="Wrap" Foreground="White" FontSize="18"/> | Thuộc SlideMenuStackPanel, phục vu cho chức năng hiển thị thông tin. Nhiệm vụ hiển thị thông tin chi tiết về Layer được người dùng chỉ định. |
| 8 | SlidePanel\_  txtBoxFind | TextBox | <TextBox x:Name="SlidePanel\_txtBoxFind" TextWrapping="Wrap" VerticalAlignment=  "Top" Margin="10,45,10,0" Height="17" Visibility="Hidden" Foreground="White" TextChanged=  "SlidePanel\_txtBoxFind\_TextChanged" TextInput="SlidePanel\_txtBoxFind\_TextInput" Background="{x:Null}"/> | Thuộc SlideMenuStackPanel, phục vụ cho chức năng tìm kiếm. Là nơi người dùng nhập vào tên Layer cần tìm kiếm. |
| 9 | SlidePanel\_  lvListLayers | ListView | <ListView x:Name=" SlidePanel\_lvListLayers " HorizontalAlignment="Left" Height="605" VerticalAlignment="Top" Width="228" Margin="10,67,0,0" Visibility="Hidden" Background="{x:Null}" Foreground="White" BorderBrush="{x:Null}"  SelectionChanged=  "SlidePanel\_lvListLayers\_SelectionChanged">  <ListView.View>  <GridView>  <GridViewColumn Header="Name" Width="114" DisplayMemberBinding="{Binding Name}" />  <GridViewColumn Header="Type" Width="114" DisplayMemberBinding="{Binding Type}" />  </GridView>  </ListView.View>  </ListView> | Thuộc SlideMenuStackPanel, phục vụ cho chức năng tìm kiếm. Là nơi chương trình trả về kết quả tìm kiếm cho người dùng. |

## Thiết kế lớp

### Thiết kế lớp cho các Layer của TensorFlow Layers API

#### Tổng quan

Xem xét trong phạm vi phần mềm sẽ xây dựng, ngoài các class đặc thù phục vụ cho các chức năng và sự vận hành của chương trình, do sự hạng chế về thời gian và kiến thức của chúng em, phần mềm này chỉ cài đặt và hiện thực hoá 13 trong tổng số 203 Layer trong Layers API, cũng như chúng em sẽ chỉ cài đặt cho các Layer này các thuộc tính quan trọng chứ không cài đặt toàn bộ tất cả các thuộc tính cho chúng.

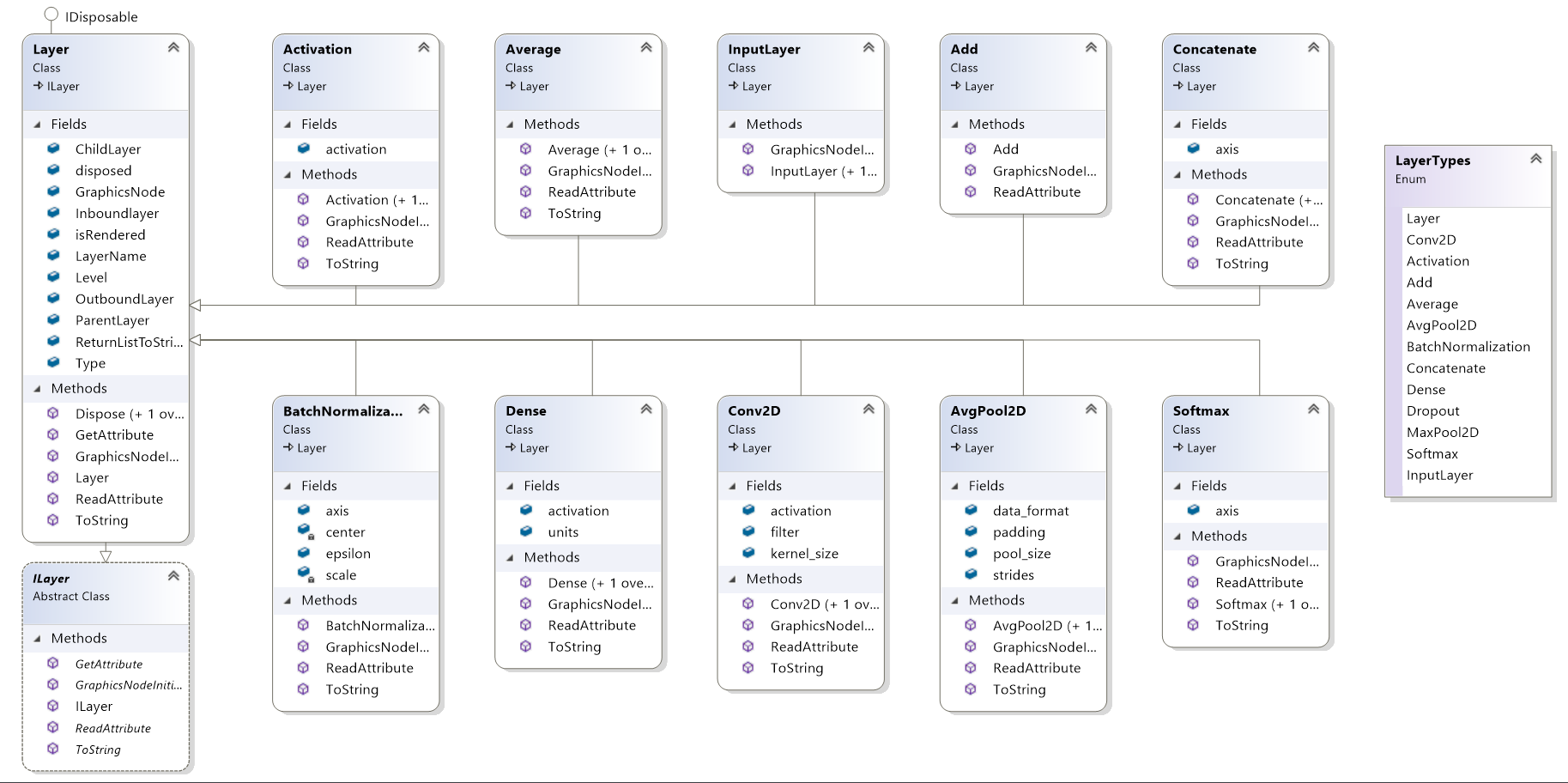
Bảng 4: Các Class và Attributes ứng với các layer.

|  |  |  |  |
| --- | --- | --- | --- |
| Class | Attributes | Kế thừa | Mô tả |
| Layer | string name |  | Layer nền, là dạng chung tất cả các Layers trong mô hình TensorFlow, nơi thực thi các phép toán cơ bản trên mạng như tích chập, định mức lô, … |
| Conv2D | int filters, int[] kernel\_size, string Activation | Layer |  |
| Activation | string activation | Layer |  |
| Add |  | Layer | Đầu vào có ít nhất 2 phần tử |
| Average |  | Layer | Đầu vào có ít nhất 2 phần tử |
| AvgPool2D | int[] pool\_size, int[] strides, string padding, string data\_format | Layer | Nếu input truyền vào strides là null thì strides sẽ tự động lấy giá trị từ pool\_size gán cho chính nó |
| BatchNormalization | int axis, float epsilon, bool center, bool scale | Layer |  |
| Concatenate | Int axis | Layer | Thuộc tính axis có thể null |
| Dense | positive int units, string activation | Layer |  |
| Dropout | float rate int noise\_shape, int seed | Layer | 0 < rate < , seed có thể null |
| MaxPool2D | int[] pool\_size, int[] strides, enum padding, string data\_format | Layer | Nếu input truyền vào strides là null thì strides sẽ tự động lấy giá trị từ pool\_size gán cho chính nó |
| Softmax | int axis | Layer |  |
| InputLayer |  | Layer | Đầu vào cho cả mạng thần kinh nhân tạo |

Bảng 5: Chi tiết chức năng các layer trong ANN Model

|  |  |  |
| --- | --- | --- |
| TT | Tên lớp | Chức năng |
| 1 | Layer | Base layer, là dạng chung của tất cả các class của Keras Layers trong mô hình TensorFlow. |
| 2 | InputLayer | Dùng làm điểm vào cho mạng (Biểu đồ các Layer). |
| 3 | Conv2D | Tạo ra một *convolution kernel,* kết hợp với Input của Layer, tạo ra một Tensor đầu ra. |
| 4 | Activation | Áp dụng hàm activation vào một Output. |
| 5 | Add | Thực hiện phép cộng tất cả các phần tử đồng dạng trong Input, cho ra Output duy nhất 1 phần tử (Đồng dạng với Input) . |
| 6 | Average | Thực hiện phép trung bình tất cả các phần tử đồng dạng trong Input, cho ra Output duy nhất 1 phần tử(Đồng dạng với Input). |
| 7 | AvgPool2D | Thực hiện phép trung bình gộp lên dữ liệu Spatial. |
| 8 | BatchNormalization | Chuẩn hoá Activation của lớp trước ở từng lô, tức là áp dụng một chuyển đổi duy trì giá trị trung bình Activation gần 0 và tiêu chuẩn Activation lệch gần 1. |
| 9 | Concatenate | Nhận vào đầu vào là một danh sách các Tensor, tất cả đều đồng dạng nhau trừ trục nối, trả về một Tensor duy nhất là nối của tất cả đầu vào |
| 10 | Dense | Là lớp nối mật độ trong Neural Network. |
| 11 | Propout | Áp dụng dropout lên input. |
| 12 | MaxPool2D | Phép tổng hợp cực đại cho dữ liệu spatial. |
| 13 | Softmax | Hàm Softmax Activation. |

#### Chi tiết



Hình 7: UML Diagram biểu diễn các lớp trong phần mềm.

Xét thấy giữa các layer có các tính chất chung, bao gồm:

* + Tên layer.
  + Loại layer.
  + Input layer.
  + Output layer.

Đó là chưa kể giữa chúng còn có các phương thức và chứa các đối tượng giống nhau (ví dụ như phương thức đọc input, đối tượng đồ hoạ, phương thức biểu diễn đối tượng về dạng chữ,…). Vì thế, để tận dụng sức mạnh của lập trình hướng đối tượng, chúng em quyết định thiết kế class Layer, class này là base cho các class khác, chứa tất cả những phương thức, thuộc tính chung của các layer. Đồng thời, các class của các layer khác kế thừa nó, bổ sung các phương thức, đối tượng của riêng nó để đáp ứng chức năng, nhiệm vụ của layer mà class đó hiện thực.

#### Đặc tả lớp

Bảng 6: Danh mục các lớp cho các Layer của TensorFlow Layers API

|  |  |  |  |
| --- | --- | --- | --- |
| TT | Tên lớp | Mục đích | SV phụ trách |
| 1 | ILayer | Định nghĩa thuộc tính và hành vi chung của các class Layer, giúp việc xây dựng các class của Layers API trở nên thống nhất. | Hoàng Vương |
| 2 | Layer  Kế thừa từ: ILayer | Base class, đối tượng hoá Layer *Layer* trong TensorFlow Model. Chứa tất cả những phương thức, thuộc tính chung của các Layer, cho các class khác sử dụng. | Hoàng Vương |
| 3 | InputLayer  Kế thừa từ: Layer | Đối tượng hoá Layer *InputLayer* trong TensorFlow Model. | Hoàng Vương |
| 4 | Conv2D  Kế thừa từ: Layer | Đối tượng hoá Layer *Conv2D* trong TensorFlow Model. | Hoàng Vương |
| 5 | Activation  Kế thừa từ: Layer | Đối tượng hoá Layer *Activation* trong TensorFlow Model. | Hoàng Vương |
| 6 | Add  Kế thừa từ: Layer | Đối tượng hoá Layer *Add* trong TensorFlow Model. | Hoàng Vương |
| 7 | Average  Kế thừa từ: Layer | Đối tượng hoá Layer *Average* trong TensorFlow Model. | Hoàng Vương |
| 8 | AvgPool2D  Kế thừa từ: Layer | Đối tượng hoá Layer *AveragePooling2D* trong TensorFlow Model. | Hoàng Vương |
| 9 | BatchNormalization  Kế thừa từ: Layer | Đối tượng hoá Layer *BatchNormalization* trong TensorFlow Model. | Hoàng Vương |
| 10 | Concatenate  Kế thừa từ: Layer | Đối tượng hoá Layer *Concatenate* trong TensorFlow Model. | Hoàng Vương |
| 11 | Dense  Kế thừa từ: Layer | Đối tượng hoá Layer *Dense* trong TensorFlow Model. | Hoàng Vương |
| 12 | Propout  Kế thừa từ: Layer | Đối tượng hoá Core *Layer* Propout trong TensorFlow Model. | Hoàng Vương |
| 13 | MaxPool2D  Kế thừa từ: Layer | Đối tượng hoá Layer *MaxPool2D* trong TensorFlow Model. | Hoàng Vương |
| 14 | Softmax  Kế thừa từ: Layer | Đối tượng hoá Layer *Softmax* trong TensorFlow Model. | Hoàng Vương |

#### Đặc tả các phương thức trong lớp

Bảng 7: Đặc tả các phương thức trong lớp Layer

|  |  |  |  |
| --- | --- | --- | --- |
| TT | Phương thức | Mục đích | Tên file, stt dòng khai báo |
| 1 | virtual ReadAttribute(string \_input)  input: \_input  output: None | Phương thức virtual, dùng để đọc plain text đã qua xử lí từ input của user thành dữ liệu, đưa vào các attributes của layer. | Layers/Topology/Layer.cs (73) |
| 2 | GetAttribute()  Input: None  output: List<string> | Phương thức virtual, dùng để xuất ra tất cả các thông tin về attributes của layer dưới dạng List<string> , mỗi phần tử trong list chưa tên và giá trị của nó. | Layers/Topology/Layer.cs (84) |
| 3 | ToString()  Input: None  output: List<string> | Phương thức virtual, dùng để xuất ra tất cả các thông tin về layer dưới dạng List<string> , mỗi phần tử trong list chưa tên và giá trị của nó. | Layers/Topology/Layer.cs (89) |
| 4 | GraphicsNodeInitialize()  Input: None  output: None | Phương thức virtual, khởi tạo đối tượng đồ hoạ cho layer với tên của đối tượng bằng với tên lớp đọc từ input người dùng | Layers/Topology/Layer.cs (79) |
| 5 | Layer()  Input: None  output: None | Khới tạo một class Layer mới, tất cả các attribute và properties cơ bản được khởi tạo về Null | Layers/Topology/Layer.cs (40) |

### Thiết kế lớp chức năng

#### Tổng quan

Ngoài các lớp chủ đạo của TensorFlow Layers API ra, để chương trình có thể hoạt động được, cần xây dựng các lớp chức năng cho chương trình. Các lớp chức năng này sẽ đảm nhiệm các công việc gồm:

* + Đọc, chuẩn hoá, soát lỗi và xử lí input từ người dùng.
  + Điều khiển, kiểm soát việc dựng các đối tượng đồ hoạ lên màn hình người dùng.

#### Đặc tả lớp

Bảng 8: Đặc tả các lớp chức năng

|  |  |  |  |
| --- | --- | --- | --- |
| TT | Tên lớp | Mục đích | SV phụ trách |
| 1 | InputHander | * Đọc đoạn Python Scripts chứa mô tả về ANN bằng Layers API mà người dùng nhập vào theo từng dòng(Mỗi dòng chứa thông tin cho Layer) * Chuẩn hoá, xử lí, tạo object layer ứng với layer được miêu tả, truyền vào thông số chính như loại layer, tên layer, đầu vào, đầu ra, các thuộc tính(dưới dạng plain text thô, chưa qua xử lí) và thêm layer đó vào Model | Hoàng Vương |
| 2 | TensorModel | Tạo ra một model ANN hoàn chỉnh bằng Layers API | Hoàng Vương |
| 3 | Render\_MasterControl | Điều khiển chung toàn bộ các tác vụ liên quan đến đồ hoạ và dựng hình trong chương trình | Hoàng Vương |
| 4 | SlidePanel\_Control | Lớp tĩnh, điều khiển riêng, đặc biệt tới các tác vụ liên quan đến đồ hoạ và dựng hình trong chương trình của đối tượng SlidePanel | Hoàng Vương |
| 5 | ConnectorRender\_Control | Điều khiển riêng, đặc biệt tới các tác vụ liên quan đến đồ hoạ và dựng hình trong chương trình các đối tượng liên kết giữa các layer | Hoàng Vương |
| 6 | Arrow | Tạo ra đối tượng đồ hoạ dạng mũi tên | Hoàng Vương |

#### Đặc tả các phương thức trong lớp

Bảng 9: Đặc tả các phương thức trong lớp Render\_MasterControl

|  |  |  |  |
| --- | --- | --- | --- |
| TT | Phương thức | Mục đích | Tên file, tt dòng khai báo |
| 1 | Render\_MasterControl(Canvas \_maincanvas, TensorModel \_model)  input: \_maincanvas, \_model  output: None | Phương thức khởi tạo của đối tượng Render\_MasterControl | RenderControl/  Render\_MasterControl.cs (30) |
| 2 | void GetParent(ref List<Layer> \_Layers)  Input: \_Layer  output: None | Phương thức virtual, dùng để xuất ra tất cả các thông tin về attributes của layer dưới dạng List<string> , mỗi phần tử trong list chưa tên và giá trị của nó. | RenderControl/  Render\_MasterControl.cs (50) |
| 3 | void GetChild(ref List<Layer> \_Layers)  Input: \_Layers  output: None | Phương thức virtual, dùng để xuất ra tất cả các thông tin về layer dưới dạng List<string> , mỗi phần tử trong list chưa tên và giá trị của nó. | RenderControl/  Render\_MasterControl.cs (38) |
| 4 | void LayerRender(TensorModel \_model)  Input: \_model  output: None | Phương thức virtual, khởi tạo đối tượng đồ hoạ cho layer với tên của đối tượng bằng với tên lớp đọc từ input người dùng | RenderControl/  Render\_MasterControl.cs (63) |
| 5 | protected void SetPosition(int CurrentLevel, ref List<Layer> \_layers, Canvas DisplayZone)  Input: None  output: None | Khới tạo một class Layer mới, tất cả các attribute và properties cơ bản được khởi tạo về Null | RenderControl/  Render\_MasterControl.cs (109) |

Bảng 10: Đặc tả các phương thức trong lớp TensorModel

|  |  |  |  |
| --- | --- | --- | --- |
| TT | Phương thức | Mục đích | Tên file, tt dòng khai báo |
| 1 | TensorModel(string name)  input: name  output: None | Phương thức khởi tạo của đối tượng TensorModel với tên model được truyền kèm theo | TensorModel.cs (22) |

Bảng 11: Đặc tả các phương thức trong lớp ConnectorRender\_Control

|  |  |  |  |
| --- | --- | --- | --- |
| TT | Phương thức | Mục đích | Tên file, tt dòng khai báo |
| 1 | ConnectorRender\_Control(Dictionary <string, List<Layer>> \_treelevel)  input: \_treelevel  output: None | Phương thức khởi tạo của đối tượng ConnectorRender\_Control. Đồng thời khởi tạo một cây phân lớp dạng Dictionary | RenderControl/  ConnectorRender\_Control.cs (21) |
| 2 | void CalcConnector()  Input: None  output: None | Tính toán toạ độ các mũi tên của các kết nối giữa các layer trên Canvas | RenderControl/  ConnectorRender\_Control.cs (80) |
| 3 | double FindBound(int levelSrc, int levelDes, string src, string des)  Input: leverSrc, levelDes, src, des  output: None | Tính toán toạ độ khoảng nhô ra lớn nhất về bên trái từ level src đến level des của cây phân lớp | RenderControl/  ConnectorRender\_Control.cs (119) |
| 4 | void calcLevelBound()  Input: None  output: None | Tính toán khoảng nhô ra lớn nhất ở từng level trong cây phân lớp rồi lưu lại trong List LevelBound | RenderControl/  ConnectorRender\_Control.cs (160) |
| 5 | void createConnector(double x1, double y1, double x2, double y2)  Input: x1, y1, x2, y2  output: None | Tạo các đối tượng kết nối dạng đoạn thẳng đi từ điểm có toạ độ x1, y1 đến điểm có toạ độ x2, y2 | RenderControl/  ConnectorRender\_Control.cs (173) |
| 6 | void createArrow(double x1, double y1, double x2, double y2)  Input: x1, y1, x2, y2  output: None | Tạo các đối tượng kết nối dạng đoạn thẳng một đầu có mũi tên đi từ điểm có toạ độ x1, y1 đến điểm có toạ độ x2, y2 | RenderControl/  ConnectorRender\_Control.cs (189) |

Bảng 12: Đặc tả các phương thức trong lớp SlidePanel\_Control

|  |  |  |  |
| --- | --- | --- | --- |
| TT | Phương thức | Mục đích | Tên file, tt dòng khai báo |
| 1 | static void Init\_SlidePanel\_Control (StackPanel pn, TextBlock tb, Storyboard sb, ListView lv, TextBox tbx, List<Layer> layers)  input: pn, tb, sb, lv, tbx, layers  output: None | Phương thức khởi tạo của đối tượng SlidePanel\_Control, truyền cho nó tất cả các control cần điều khiển | RenderControl/  SlidePanel\_Control.cs (44) |
| 2 | static void SlidePanel\_Show(string namelayer, SlidePanel\_Mode displayMode)  Input: namelayer  output: None | Đổi chế độ của SlidePanel về thành hiển thị thông tin chi tiết của một layer với tên cho trước | RenderControl/  SlidePanel\_Control.cs (56) |

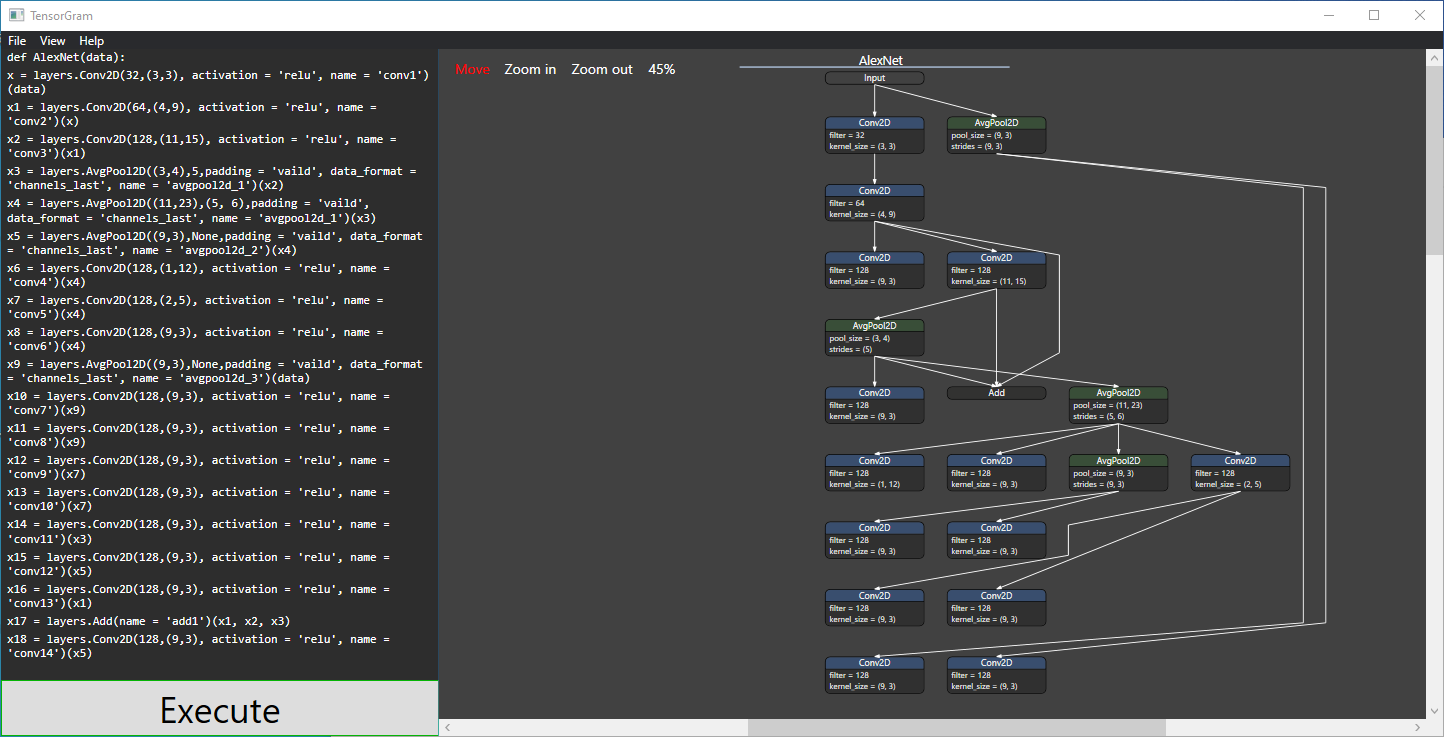
Bảng 13: Đặc tả các phương thức trong lớp Arrow

|  |  |  |  |
| --- | --- | --- | --- |
| TT | Phương thức | Mục đích | Tên file, dòng khai báo |
| 1 | static Shape DrawLinkArrow(Point p1, Point p2)  input: p1, p2  output: None | Tạo đối tượng Shape người tuỳ chỉnh dạng đoạn thẳng có một đầu là mũi tên đi từ điểm p1 đến điểm p2 | GraphicsObject/  Arrow.cs (16) |

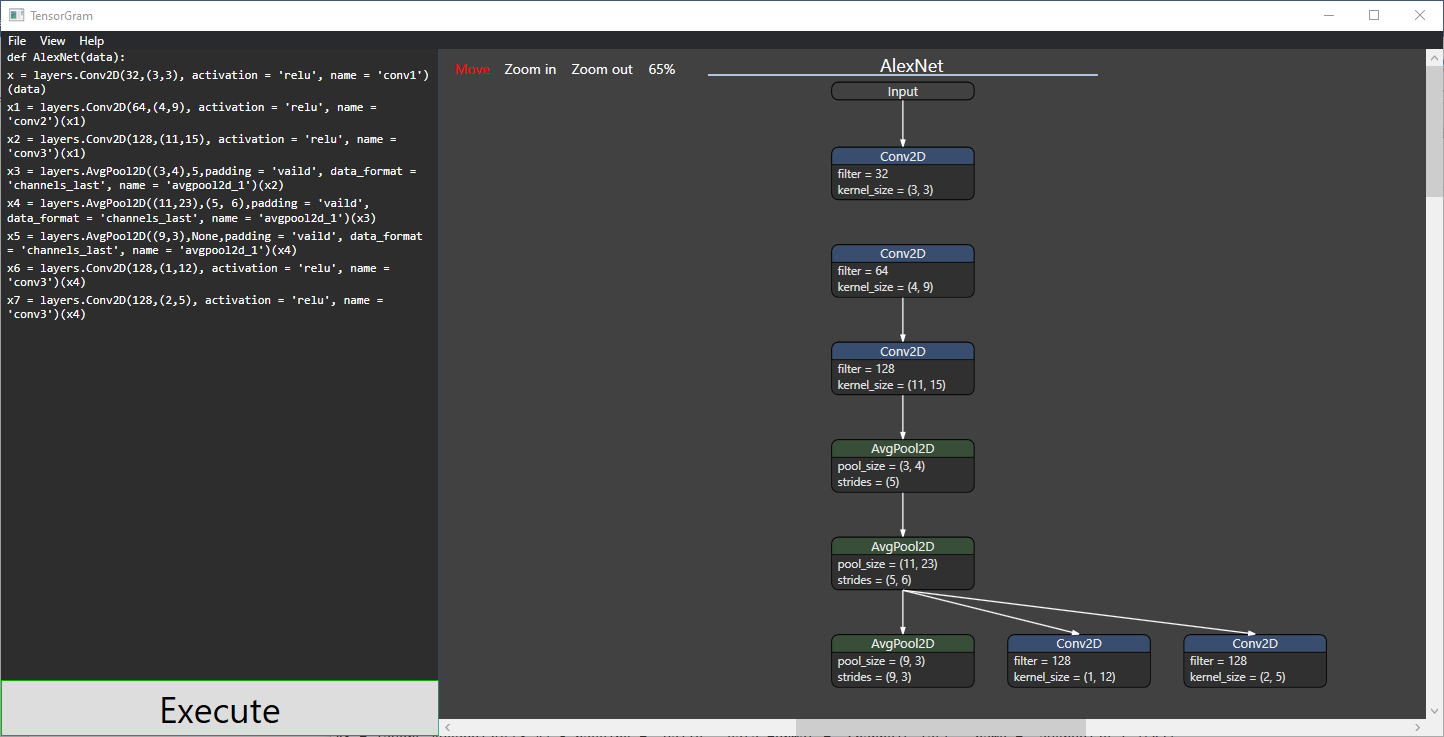
Bảng 14: Đặc tả các phương thức trong lớp TextInput\_Hander

|  |  |  |  |
| --- | --- | --- | --- |
| TT | Phương thức | Mục đích | Tên file, dòng khai báo |
| 1 | TextInput\_Hander(string textinput, ref TensorModel \_ModelOutput)  input: textinput, TensorModel\_ ModelOutput  output: None | Khởi tạo đối tượng TextInput\_Hander. Đọc và chuẩn hoá input scripts, chia cả đoạn scripts thành từng dòng, lưu vào List<string> RawTextData . Đồng thời tạo một model ANN rỗng để chứa các Layer | TextInput\_Hander.cs (15) |
| 2 | TensorModel ReadModel(List<string> \_RawData)  input: \_RawData  output: TensorModel | Từ các dòng đã được xử lí và chuẩn hoá, tạo nên một Model ANN hoàn chỉnh với chuẩn TensorFlow Layer API | TextInput\_Hander.cs (35) |
| 3 | Layer LayerReader(string \_input)  input: \_input  output: Layer | Đọc scripts theo từng dòng, trả về một đối tượng Layer theo mô tả của scripts | TextInput\_Hander.cs (61) |
| 4 | Layer CreateLayerByType(string type)  input: type  output: Layer | Trả về một Layer thô, chưa có dữ liệu có kiểu như mô tả ( Kiểu Conv2D, Avg, Add, Dense, … ) | TextInput\_Hander.cs (97) |

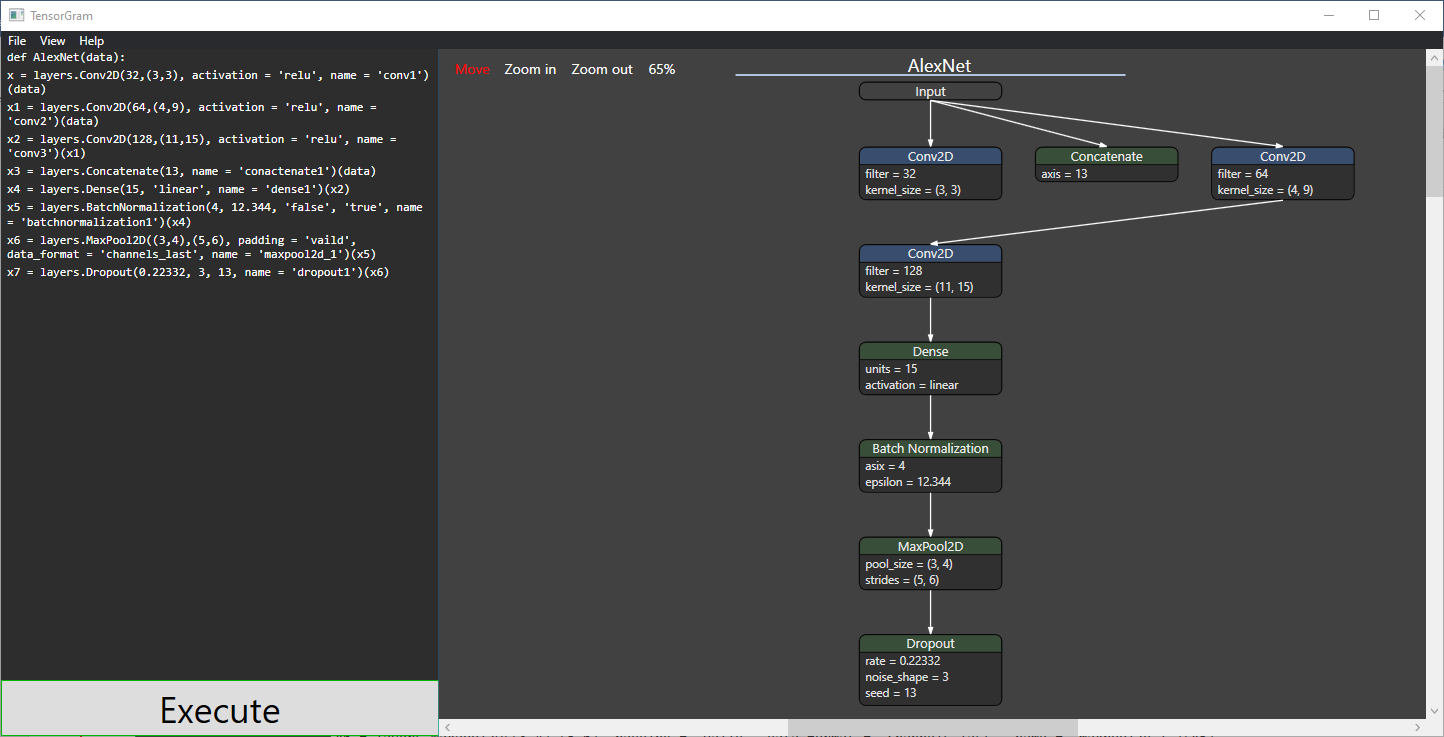
# Chương 4: Cài đặt và kiểm thử



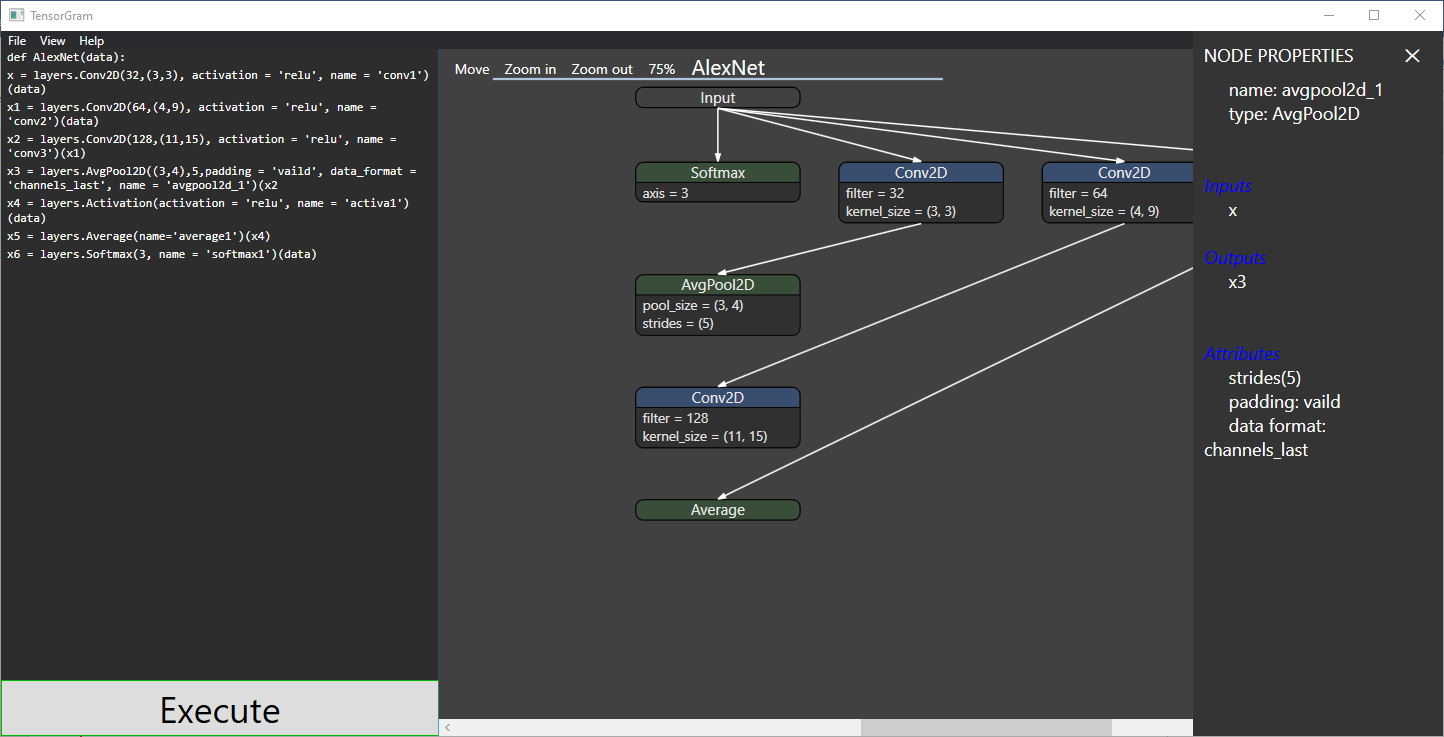
Hình 8: Kiểm thử 1



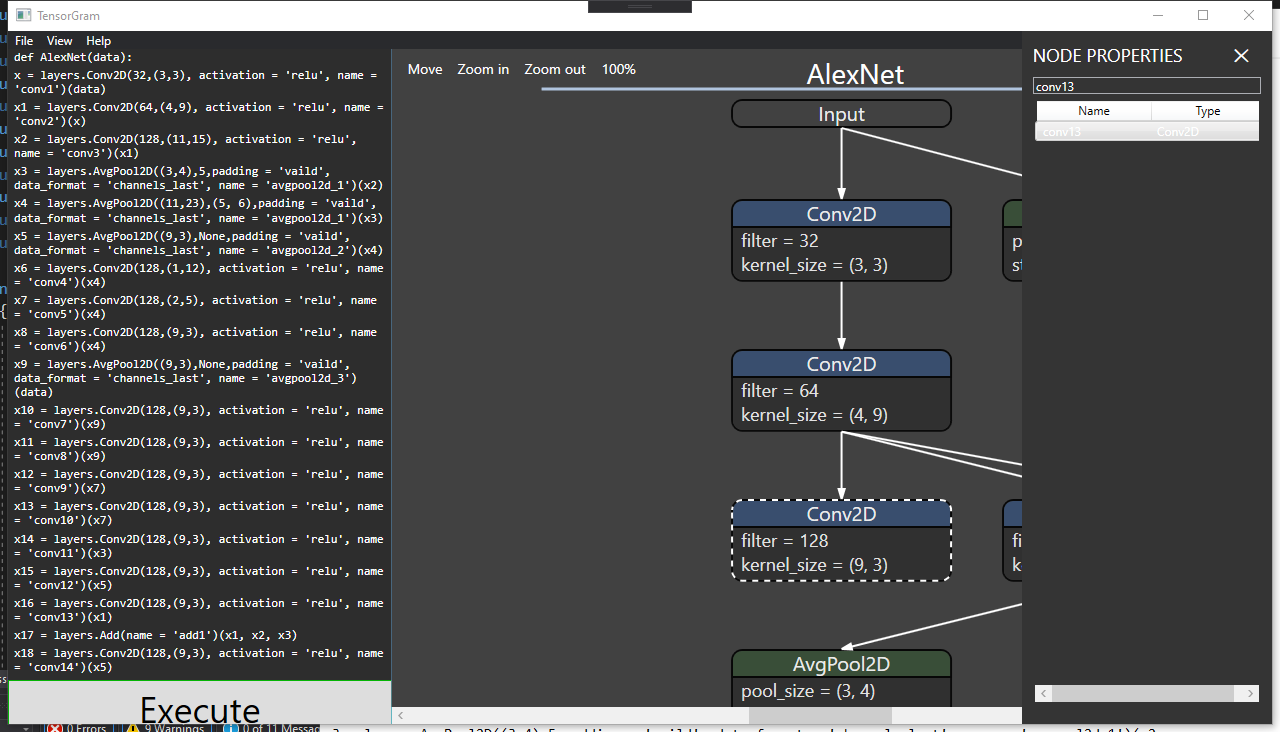
Hình 9: Kiểm thử 2



Hình 10: Kiểm thử 3



Hình 11: Kiểm thử 4



Hình 12: Kiểm thử 5

# Chương 5: Kết luận và hướng phát triển

## Kết luận

Về cơ bản, nhóm tự nhận xét phần mềm của nhóm đã giải quyết được được 95% yêu cầu mà đồ án đặt đặt ra. Sau đây là ưu điểm cũng như tồn tại của phần mềm .

* Ưu điểm:
  + Giao diện gọn gàng, dễ tiếp cận, dễ làm quen.
  + Dung lượng khá nhẹ (Chỉ 117KB cho một file exe duy nhất).
  + Chương trình tốn rất ít tài nguyên hệ thống khi hoạt động.
  + Chương trình chạy ổn định, cho ra kết quả chính xác, không bị crash trong quá trình thực thi yêu cầu người dùng.
* Nhược điểm:
  + Diagram xuất ra chưa đẹp mắt (Chưa sắp xếp được các Layer con nằm ngay ngắn ngay bên dưới Layer cha).
  + Thuật toán tỏ ra kém hiệu quả khi xử lí dữ liệu đầu vào lớn (Xử lí và xuất ra kết quả tốn nhiều thời gian).
  + Chưa có chức năng kiểm lỗi cho dữ liệu đầu vào.

## Hướng phát triển

* Thêm tính năng đọc dữ liệu đầu vào từ file.
* Tối ưu hoá thuật toán đối với dữ liệu đầu vào từ lớn đến rất lớn.
* Chỉnh sửa thuật toán dựng hình để có thể xếp các Layer con bên dưới Layer cha một các gọn gàng và đúng đắn nhất.
* Viết thêm tính năng kiểm lỗi dữ liệu đầu vào.
* Cải thiện giao diện người dùng.

# Tài liệu tham khảo

[1]. Keras Sharp – Tác giả: Cesarsouza [github.com/cesarsouza/keras-sharp](https://github.com/cesarsouza/keras-sharp)

[2]. Tài liệu của TensorFlow về Layers API <https://www.tensorflow.org/versions/r1.15/api_docs/python/tf/layers?hl=fa>

[3]. Bắt đầu với Machine Learning thông qua Tensorflow - Tác giả: Trần Đức Tâm <https://www.kipalog.com/posts/Bat-dau-voi-Machine-Learning-thong-qua-Tensorflow--Phan-I-2>

[4]. Loạt video TensorFlow.js: Layers API Part 1 & Part 2 – Tác giả: The Coding Train

<http://www.youtube.com/watch?v=F4WWukTWoXY>

[5]. Tài liệu của Microsoft về Windows Presentation Foundation

<https://docs.microsoft.com/en-us/dotnet/framework/wpf/>

[6]. Mã nguồn NetScope <https://github.com/ethereon/netscope>

[7]. Tài liệu về TensorBoard <https://www.tensorflow.org/tensorboard>

[8]. Mã nguồn phần mềm Netron – Tác giả: Lutz Roeder <https://github.com/lutzroeder/netron>