

**NGÀNH CÔNG NGHỆ THÔNG TIN**

**ĐỒ ÁN 1**

**KẾ HOẠCH THỰC HIỆN**

**PHẦN MỀM TENSORGRAM**

|  |  |
| --- | --- |
| Nhóm sinh viên thực hiện: | |
| Huỳnh Quốc Hoàng Vương | 17110256 |
| Minh Trung  Việt | 17110  17110 |

Giảng viên hướng dẫn: ThS. Huỳnh Xuân Phụng

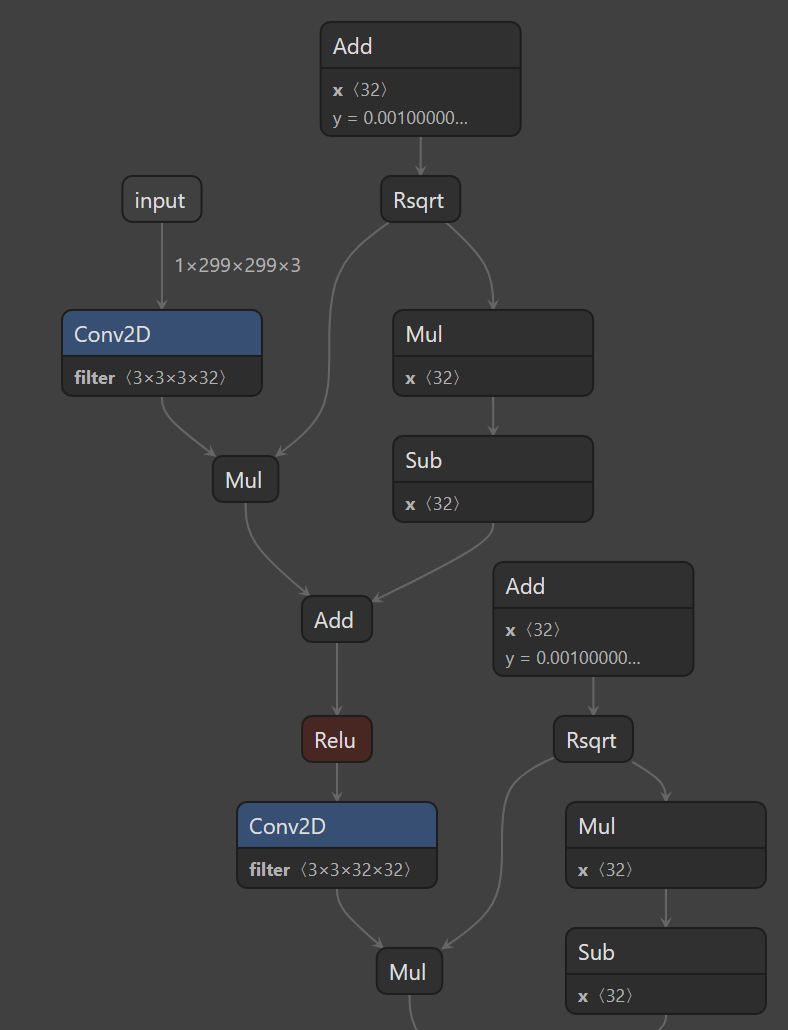
Tp. Hồ Chí Minh, tháng 9 năm 2019

1. **Nghiên cứu, phân tích yêu cầu đề tài**
   1. ***Tìm hiểu về Tensorflow***

TensorFlow là một thư viện phần mềm mã nguồn mở dành cho máy học trong nhiều loại hình tác vụ nhận thức và hiểu ngôn ngữ. Nó hiện đang được sử dụng cho cả nghiên cứu lẫn sản xuất bởi 50 đội khác nhau trong hàng tá sản phẩm thương mại của Google, như nhận dạng giọng nói, Gmail, Google Photos, và tìm kiếm, nhiều trong số đó đã từng sử dụng chương trình tiền nhiệm DistBelief của nó. TensorFlow nguyên thủy được phát triển bởi đội Google Brain cho mục đích nghiên cứu và sản xuất của Google và sau đó được phát hành theo giấy phép mã nguồn mở Apache 2.0 vào ngày 9/11/2015.

TensorFlow thế hệ thứ hai của hệ thống học máy của Google Brain, với một bản cài đặt tham khảo đã phát hành dưới dạng phần mềm mã nguồn mở vào ngày 9/11/2015. Trong khi bản cài đặt tham khảo chạy trên một thiết bị đơn, TensorFlow có thể chạy trên nhiều CPU và GPU (với nhiều mở rộng CUDA tùy chọn cho việc tính toán đa năng trên các GPU). Nó chạy trên desktop Linux hoặc Mac OS X 64-bit hoặc các hệ thống máy chủ, cũng như trên các nền tảng điện toán di động, bao gồm Android và iOS của Apple. Các tính toán của TensorFlow được thể hiện dưới dạng các biểu đồ dataflow chi tiết. Nhiều nhóm tại Google đã chuyển từ DistBelief sang TensorFlow để phục vụ cho việc nghiên cứu và sản xuất. Thư viện thuật toán này bắt nguồn từ nhu cầu của Google để hướng dẫn (lập trình) các hệ thống máy tính, được gọi là mạng nơron, để tìm hiểu và lý luận tương tự cách thức của con người, vì vậy mà các ứng dụng mới có thể được xuất phát từ đây có thể đảm nhận các vai trò và chức năng vốn trước đây chỉ dành cho những người có khả năng; cái tên TensorFlow bản thân nó xuất phát từ các thao tác mà các mạng nơron như vậy thực hiện trên các mảng dữ liệu đa chiều. Những mảng đa chiều được gọi là các "tensor" nhưng khái niệm này là không giống với khái niệm 'tensor' trong toán học. Mục đích là để huấn luyện các mạng nơron phát hiện và giải mã các mẫu và các mối tương quan.

* 1. ***TensorFlow Keras Layers***



*Hình 1: Model TensorFlow biểu diễn bằng Keras Layer*

Một TensorFlow model được định nghĩa bằng nhiều Layer. TensorFlow model nhận dữ liệu đầu vào của người dùng, đưa dữ liệu đi qua các Layer bên trong nó. Các Layer này được kết nối với nhau thực hiện các hoạt động trong mạng neural (như tích chập, cộng, …), các hoạt động này yêu cầu quản lý trọng lượng, tổn thất, cập nhật và kết nối giữa các lớp. Cuối cùng, người dùng nhận lại được dữ liệu đã qua xử lí. Để sử dụng TensorFlow model, người dùng cần định nghĩa và cài đặt các lớp trong nó.

* 1. ***Phân tích đề tài***

Đề tài yêu cầu xây dựng phần mềm trên Windows có giao diện đồ hoạ người dung rõ ràng. Phần mềm nhận input từ người dùng là đoạn code python nội dung mô tả một Tensorflow Model, đầu ra là một flowchart mô tả Tensorflow Model trực quan cho người dùng.

1. **Xác định ngôn ngữ lập trình và công cụ thực hiện phần mềm theo yêu cầu của đề tài**

* Sử dụng ngôn ngữ C# trên nền .NET 4.7
* Sử dụng WPF (Windows Presentation Foundation) API để thiết kế giao diện người dùng (User Interface)
* Công cụ hỗ trợ thực hiện phần mềm:
  + Microsoft Visual Studio 2019
  + Blend for Visual Studio 2019
  + Adobe Illustrator CC 2019
  + GitHub for Desktop

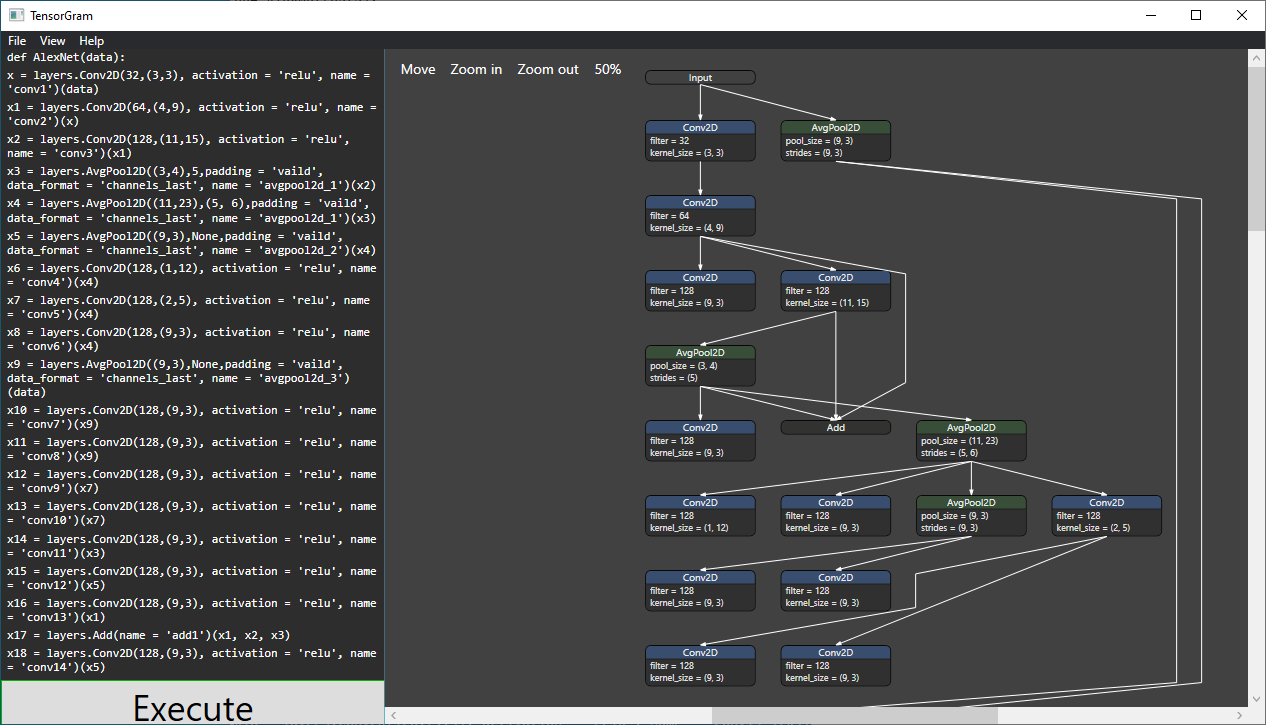
1. **Định hướng thiết kế phần mềm**
   1. ***Hướng thiết kế***

Thiết kế phần mềm hướng đối tượng. Một Model TensorFlow là một đối tượng trong chương trình (Object Model). Một Model TensorFlow lại có nhiều lớp (Layer) tương ứng với một Object Model trong chương trình lại chứa 2 hoặc nhiều đối tượng Layer (Object Layer) có thuộc tính rõ ràng. Các đối tượng này được định nghĩa bởi người dùng thông qua đoạn code python mà người dùng nhập vào trong giao diện làm việc của chương trình.

* 1. ***Các Control dự kiến sử dụng***

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên Control** | **Hướng sử dụng** |
| 1 | RichTextBox | Là nơi người dùng nhập input dưới dạng pain text là một đoạn code Python định nghĩa TensorFlow Model. |
| 2 | Button | Ghi nhận và thực hiện các thao tác bằng chuột của người dùng. |
| 3 | TextBlock | Hiển thị thông tin về một Layer do người dùng chỉ định trong TensorFlow Model dưới dạng pain text. |
| 4 | Panel | Chứa, định hình cho các Control khác, đồng thời quy định, phân chia các khu vực chức năng khác nhau trong cửa sổ làm việc của phần mềm |
| 5 | Canvas | Khu vực chính của phần mềm, nơi phần mềm xuất những thông tin về Model dưới dạng đồ hoạ (Bao gồm tất cả các lớp, thông tin cơ bản của các lớp, các connect giữa chúng, …) sau khi đã tính toán và xử lí thông tin người dùng nhập vào RichTextBox. |
| 6 | Label | Hiển thị một số thông tin thứ yếu dưới dạng pain text, ngoài ra có thể dùng làm như một button |
| 7 | User Control | Dùng để thiết kế đối tượng đồ hoạ cho các Layer |
| 8 | Menu |  |
| 9 | ScrollViewer |  |

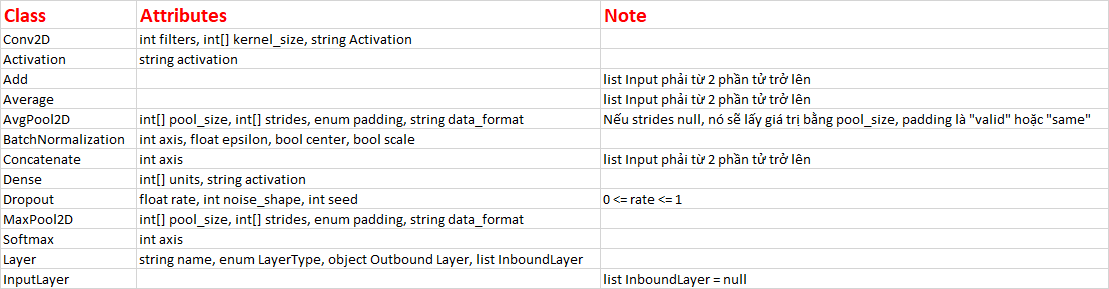
*Bảng 1: Danh sách các Control sử dụng trong phần mềm.*



*Hình 2: Giao diện người dùng dự kiển của phần mềm.*

* 1. ***Thiết kế lớp***
     1. *Tổng quan*

Xem xét trong phạm vi phần mềm sẽ xây dựng, có rất nhiều class được viết và thiết kế để xử lí các phần khác nhau cho các công việc khác nhau để xử lí dữ liệu đầu vào và xuất ra dữ liệu qua xử lí đáp ứng nhu cầu người dùng. Nhưng vì tính chất của văn bản nên mục này chúng em chỉ sẽ đề cập và đặc tả đến những class cốt lõi xây dựng nên chức năng chính của phần mềm, đó là các class Keras Layer của TensorFlow model. Nhưng 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 Keras Layer, cũng như chúng em sẽ chỉ cài đặt cho các layer này một số attributes tiêu biểu chứ không cài đặt toàn bộ tất cả attributes cho chúng.

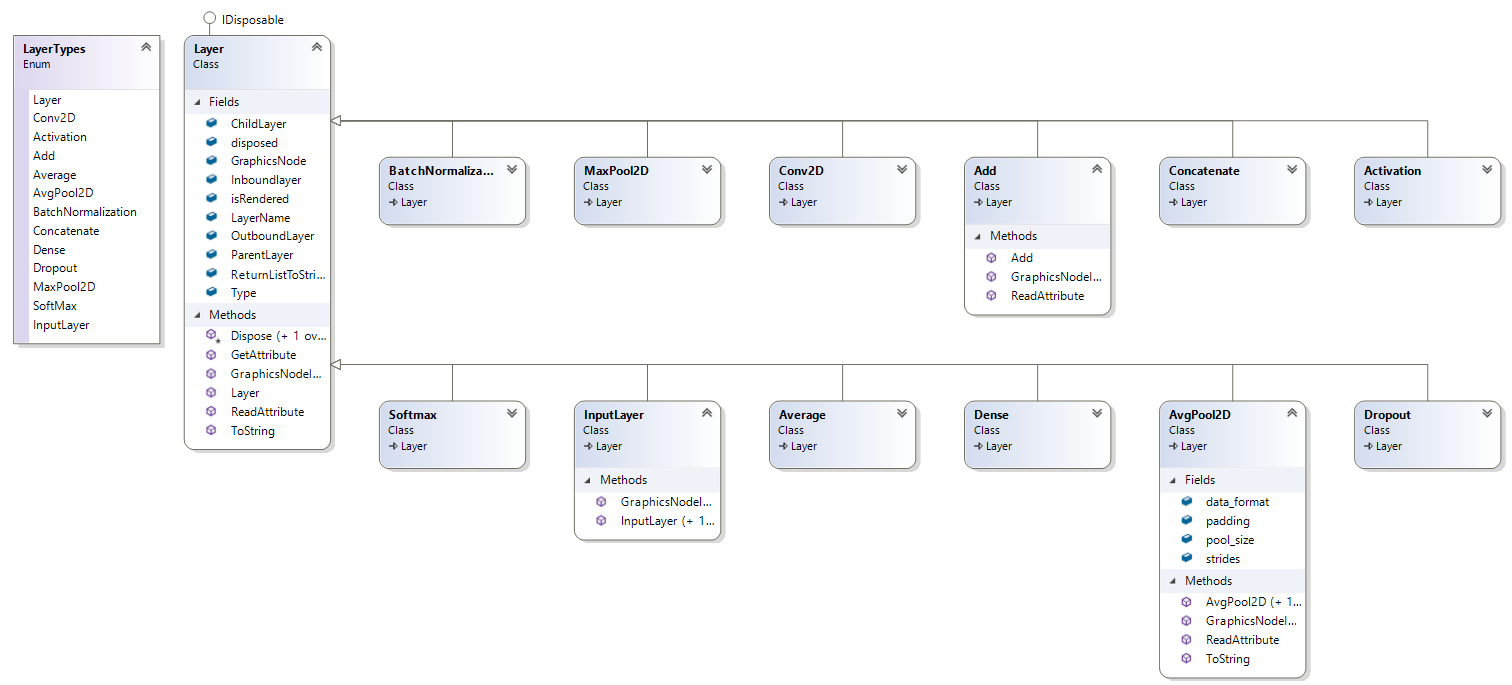


*Hình 3: Các Class và Attributes ứng với các layer.*

|  |  |  |
| --- | --- | --- |
| STT | 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 | Layer được sử dụng làm điểm vào trong Mạng (Biểu đồ các Layer). |
| 3 | Conv2D | Layer này tạo ra một ***convolution kernel*** được tích hợp với Input của Layer, tạo ra một Tensor đầu ra. |
| 4 | Activation | Áp dụng hàm activation vào một Output. |
| 5 | Add | Layer này cộng tất cả các phần tử đồng dạng trong Input, cho ra Output duy nhất 1 phần tử(Đồng dạng với Input) |
| 6 | Average | Layer này 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 | Layer này trung bình gộp cho dữ liệu **spatial** |
| 8 | BatchNormalization |  |
| 9 | Concatenate |  |
| 10 | Dense |  |
| 11 | Propout |  |
| 12 | MaxPool2D |  |
| 13 | Softmax |  |

*Bảng 2: Chi tiết chức năng các layer trong TensorFlow Model*

* + 1. Thiết kế



*Hình 4 : UML Diagram biểu diễn các lớp sẽ xây dựng 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 methods và chứa các đối tượng giống nhau (ví dụ như method đọc input, đối tượng đồ hoạ, overdrive method .ToString(), …). 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 Method, đối tượng của riêng nó để đáp ứng chức năng, nhiệm vụ của layer mà class đó hiện thực.

* + 1. *Đặc tả lớp*

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên lớp | Mục đích | Các SV phụ trách |
| 1 | Layer  Kế thừa từ: None | 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 |
| 2 | InputLayer  Kế thừa từ: Layer | Đối tượng hoá Layer ***InputLayer*** trong TensorFlow Model. | Hoàng Vương |
| 3 | Conv2D  Kế thừa từ: Layer | Đối tượng hoá Layer ***Conv2D*** trong TensorFlow Model. | Hoàng Vương |
| 4 | Activation  Kế thừa từ: Layer | Đối tượng hoá Layer ***Activation*** trong TensorFlow Model. |  |
| 5 | Add  Kế thừa từ: Layer | Đối tượng hoá Layer ***Add*** trong TensorFlow Model. | Hoàng Vương |
| 6 | Average  Kế thừa từ: Layer | Đối tượng hoá Layer ***Average*** trong TensorFlow Model. |  |
| 7 | AvgPool2D  Kế thừa từ: Layer | Đối tượng hoá Layer ***AveragePooling2D*** trong TensorFlow Model. | Hoàng Vương |
| 8 | BatchNormalization  Kế thừa từ: Layer | Đối tượng hoá Layer ***BatchNormalization*** trong TensorFlow Model. |  |
| 9 | Concatenate  Kế thừa từ: Layer | Đối tượng hoá Layer ***Concatenate*** trong TensorFlow Model. |  |
| 10 | Dense  Kế thừa từ: Layer | Đối tượng hoá Layer ***Dense*** trong TensorFlow Model. |  |
| 11 | Propout  Kế thừa từ: Layer | Đối tượng hoá Core ***Layer*** Propout trong TensorFlow Model. |  |
| 12 | MaxPool2D  Kế thừa từ: Layer | Đối tượng hoá Layer ***MaxPool2D*** trong TensorFlow Model. |  |
| 13 | Softmax  Kế thừa từ: Layer | Đối tượng hoá Layer ***Softmax*** trong TensorFlow Model. |  |

*Bảng 3: Danh mục các lớp*

* + 1. *Đặc tả các phương thức trong lớp*

|  |  |  |  |
| --- | --- | --- | --- |
| 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) |

*Bảng 4: Đặc tả các phương thức trong lớp Layer*