**Dàn ý Báo Cáo: Kiến trúc phần cứng cho Học sâu (Deep Learning)**

**I. Giới thiệu**

1. **Định nghĩa và ý nghĩa của Học sâu**  
   Học sâu là một nhánh của **Trí tuệ nhân tạo (Artificial Intelligence - AI)**, trong đó các thuật toán được lấy cảm hứng từ cách hoạt động của não bộ con người. Nó sử dụng **mạng nơ-ron sâu (Deep Neural Networks - DNNs)**, bao gồm nhiều tầng (layers) tính toán, để tự động trích xuất và học các đặc điểm phức tạp từ dữ liệu.

**Nền tảng cốt lõi:**

* Các đơn vị tính toán cơ bản là **nơ-ron**, mô phỏng cách tế bào thần kinh hoạt động trong não.
* Sử dụng trọng số (weights) và chức năng kích hoạt (activation functions) để điều chỉnh ảnh hưởng giữa các tầng trong mạng nơ-ron.

1. **Ý nghĩa của học sâu**

* **Tự động hóa quá trình học**: Học sâu cho phép máy tính tự động nhận diện và học hỏi các đặc điểm từ dữ liệu, giống như cách một đứa trẻ học hỏi từ kinh nghiệm mà không cần phải chỉ ra từng chi tiết cụ thể.
* **Xử lý dữ liệu phức tạp**: Học sâu có khả năng xử lý nhiều loại dữ liệu phức tạp như hình ảnh, âm thanh và văn bản, giúp tạo ra các ứng dụng AI mạnh mẽ trong nhiều lĩnh vực như nhận diện khuôn mặt, xử lý ngôn ngữ tự nhiên và xe tự lái.
* **Yêu cầu dữ liệu lớn**: Để đạt được hiệu suất tốt nhất, các mô hình học sâu thường cần một lượng lớn dữ liệu để huấn luyện, điều này phản ánh sự phát triển của công nghệ số và sự bùng nổ dữ liệu hiện nay.

Học sâu không chỉ giúp máy móc học hỏi và thực hiện các nhiệm vụ phức tạp mà còn trở thành công cụ chính để giải quyết những thách thức lớn trong khoa học và công nghệ.

**A diagram of a deep learning network

Description automatically generated**

**II. Ứng dụng của Học sâu**

* **Thị giác máy tính**: Phân loại ảnh, nhận diện vật thể, phân đoạn ảnh.
* **Xử lý ngôn ngữ tự nhiên**: Dịch thuật, tổng hợp giọng nói, nhận diện giọng nói.
* **Y tế**: Phát hiện ung thư, gấp protein, nghiên cứu thuốc.
* **Trò chơi**: AlphaGo và AlphaStar trong các trò chơi chiến lược.

A collage of a computer vision

Description automatically generated

**III. Khó khăn và thách thức**

1. **Yêu cầu dữ liệu lớn**: Cần có lượng lớn dữ liệu chất lượng cao để huấn luyện, điều này có thể khó khăn trong việc thu thập.
2. **Chi phí tính toán cao**: Đào tạo mô hình yêu cầu sức mạnh tính toán lớn, dẫn đến chi phí cao cho phần cứng và thời gian.
3. **Thiếu khả năng diễn giải**: Nhiều mô hình hoạt động như "hộp đen", khó hiểu cách đưa ra quyết định, gây thiếu tin tưởng.
4. **Vấn đề overfitting**: Mô hình có thể học quá mức từ dữ liệu huấn luyện, dẫn đến hiệu suất kém trên dữ liệu mới.
5. **Khó khăn trong duy trì và cập nhật**: Cần thường xuyên đào tạo lại mô hình để duy trì độ chính xác, tốn thời gian và nguồn lực.

**IV. Thách thức và cơ hội**

* Yêu cầu hiểu biết về miền, các biến thể của thuật toán (xác định những thuật toán nào quan trọng) và khối lượng công việc kết quả.
* Xử lý tính toán không đồng nhất ở cấp độ hệ thống.
* Xử lý các thiết bị không đồng nhất (công nghệ thiết bị mới nổi).
* Có thể có các hạn chế tài nguyên chặt chẽ hơn vì phần cứng không thể được sử dụng cho các ứng dụng khác.
* [Đồng thiết kế giữa các thuật toán và phần cứng](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,udsedgeshop,channelstable,ntpquery,devtoolsapi,udsinwin11,udsdlpconsent,udscstart,cspgrd,&shellsig=4a21f2b7ee76c7a44ce5176182631ca48fff3866&setlang=en-US&darkschemeovr=1&udsps=0&udspp=0#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C0%7C03683a82-0bed-4afa-be09-f337a0ab7e5c).
* Ngôn ngữ miền cụ thể để lập trình phần cứng chuyên dụng.
* Công cụ để đánh giá và tạo mẫu nhanh chóng.

**V. Ứng dụng thực tế**

1. **Xe tự lái**
   * Tạo ra khối lượng dữ liệu khổng lồ và yêu cầu năng lượng lớn.
2. **Học sâu trong Y tế và Sinh học**
   * AlphaFold2 giải quyết bài toán gấp protein kéo dài hàng thập kỷ.
   * Phát hiện và điều trị COVID-19.

**VI. Dòng thời gian phát triển Deep Neural Networks(DNNs):**

* 1940s: Mạng nơ-ron (Neural Networks) lần đầu được đề xuất.
* 1960s: Ý tưởng về mạng nơ-ron sâu (Deep Neural Networks - DNNs) được giới thiệu.
* 1989: Mạng nơ-ron LeNet được phát triển để nhận dạng chữ số viết tay.
* 1990s: Phần cứng hỗ trợ mạng nơ-ron nông xuất hiện (ví dụ: Intel ETANN, 1992).
* 2011: Bước đột phá về nhận diện giọng nói dựa trên DNN (Microsoft real-time speech translation).
* 2012: DNN vượt qua các phương pháp truyền thống trong thị giác máy tính (AlexNet cho phân loại hình ảnh).
* 2014+: Sự phát triển của nghiên cứu bộ tăng tốc DNN, với các ví dụ như Neuflow, DianNao.

**VII. Các nền tảng phần cứng hỗ trợ học sâu:**

* Bộ xử lý trung tâm (CPU): Intel, ARM, AMD.
* Bộ xử lý đồ họa (GPU): NVIDIA, AMD, được sử dụng phổ biến nhờ khả năng tăng tốc tính toán.
* Mạch logic khả trình hạt mịn (FPGA): Ví dụ: Microsoft BrainWave.
* Phần cứng lập trình/khả trình hạt thô: Wave Computing, Plasticine, Graphcore.
* Phần cứng đặc thù ứng dụng: Các bộ tăng tốc như Neuflow, DianNao, Eyeriss, TPU, SCNN, v.v.

Chương 2: Tổng quan về các thành phần của Deep Neural Network

**I. Tổng quan của Deep Neural Networks:**

Deep Neural Networks (DNN) là một loại mạng nơ-ron nhân tạo (Artificial Neural Network) với nhiều lớp ẩn (hidden layers), được thiết kế để mô phỏng cách bộ não con người xử lý thông tin. DNN là thành phần chính của học sâu (Deep Learning), giúp máy tính tự động học và trích xuất các đặc trưng từ dữ liệu phức tạp như hình ảnh, âm thanh và văn bản.

**A diagram of a deep learning

Description automatically generated**

1. **Artificial Intelligence (AI):**
   * **Định nghĩa: Trí tuệ nhân tạo (AI) là lĩnh vực nghiên cứu trong khoa học máy tính nhằm phát triển hệ thống có khả năng thực hiện các nhiệm vụ mà thường cần đến trí thông minh của con người, như nhận diện hình ảnh, xử lý ngôn ngữ tự nhiên và ra quyết định.**
   * **Ý nghĩa: AI giúp tự động hóa nhiều quy trình, cải thiện hiệu suất làm việc và tạo ra các giải pháp sáng tạo cho các vấn đề phức tạp trong đời sống.**
2. **Machine Learning (ML):**
   * **Định nghĩa: Học máy là một nhánh của AI, cho phép máy tính học hỏi từ dữ liệu mà không cần lập trình cụ thể. ML sử dụng các thuật toán để phân tích dữ liệu, nhận diện mẫu và đưa ra dự đoán hoặc quyết định.**
   * **Ý nghĩa: ML giúp cải thiện khả năng tự động hóa và tối ưu hóa các nhiệm vụ, từ dự đoán giá cả đến phân loại hình ảnh, góp phần nâng cao hiệu quả trong nhiều lĩnh vực.**
3. **Brain-Inspired:**
   * **Định nghĩa: Đây là khái niệm chỉ việc thiết kế các hệ thống AI dựa trên cách thức hoạt động của bộ não con người. Các mô hình này thường cố gắng mô phỏng cấu trúc và chức năng của nơ-ron trong não.**
   * **Ý nghĩa: Việc lấy cảm hứng từ não bộ giúp phát triển các thuật toán học sâu mạnh mẽ hơn, cho phép máy tính giải quyết các vấn đề phức tạp một cách hiệu quả hơn.**
4. **Spiking Neural Networks (SNN):**
   * **Định nghĩa: SNN là một loại mạng nơ-ron nhân tạo mô phỏng cách mà nơ-ron trong não bộ giao tiếp với nhau thông qua xung điện (spikes). Thay vì sử dụng tín hiệu liên tục, SNN xử lý thông tin theo dạng xung.**
   * **Ý nghĩa: SNN có thể cung cấp một phương pháp hiệu quả hơn để xử lý thông tin thời gian thực và giảm thiểu tiêu thụ năng lượng, mở ra khả năng cho các ứng dụng AI trong các thiết bị nhúng.**
5. **Neural Networks (NN):**
   * **Định nghĩa: Mạng nơ-ron là một cấu trúc tính toán được xây dựng từ các nơ-ron nhân tạo, mô phỏng cách hoạt động của não bộ con người để xử lý thông tin. NN có thể bao gồm nhiều lớp khác nhau để học hỏi từ dữ liệu.**
   * **Ý nghĩa: Mạng nơ-ron là nền tảng cho nhiều ứng dụng học sâu, cho phép máy tính học hỏi và nhận diện mẫu từ dữ liệu phức tạp như hình ảnh và âm thanh.**
6. **Deep Learning (DL):**
   * **Định nghĩa: Học sâu là một nhánh con của học máy, sử dụng mạng nơ-ron sâu (Deep Neural Networks) với nhiều lớp ẩn để học hỏi từ dữ liệu lớn. DL đặc biệt hiệu quả trong việc xử lý dữ liệu không có cấu trúc.**
   * **Ý nghĩa: Học sâu đã cách mạng hóa nhiều lĩnh vực như nhận diện hình ảnh, xử lý ngôn ngữ tự nhiên và xe tự lái, giúp cải thiện độ chính xác và hiệu suất của các hệ thống AI hiện đại.**

**A diagram of a graph

Description automatically generated**

**II. Thuật ngữ cơ bản của DNN**

1. **Các thành phần trong DNN:**
   * **Neuron:** Là đơn vị cơ bản của mạng nơ-ron, tương tự như tế bào thần kinh trong não người. Mỗi neuron nhận đầu vào, thực hiện một phép toán và sản xuất đầu ra.
   * **Synapse (trọng số):** Là kết nối giữa các neuron, nơi mà tín hiệu được truyền từ neuron này sang neuron khác. Mỗi synapse có một trọng số (weight) quyết định mức độ ảnh hưởng của tín hiệu.
   * **Layer (tầng):** Bao gồm các trọng số liên kết đầu vào và đầu ra.
   * **Activation:** Là quá trình mà neuron quyết định xem có nên kích hoạt hay không dựa trên đầu vào nhận được. Hàm kích hoạt giúp thêm tính phi tuyến vào mô hình, cho phép mạng học các mối quan hệ phức tạp.

Các thành phần của một DNN

1. **Input Layer (Lớp Đầu Vào)**:
   * Đây là lớp đầu tiên của mạng, nơi tiếp nhận dữ liệu đầu vào. Mỗi nút trong lớp này tương ứng với một đặc trưng trong dữ liệu.
2. **Hidden Layer (Lớp Ẩn)**:
   * Các lớp này nằm giữa lớp đầu vào và lớp đầu ra. DNN có thể có nhiều lớp ẩn, mỗi lớp thực hiện các phép toán phức tạp để trích xuất các đặc trưng từ dữ liệu. Số lượng lớp ẩn và số lượng neuron trong mỗi lớp có thể thay đổi tùy thuộc vào bài toán cụ thể.
3. **Output Layer (Lớp Đầu Ra)**:
   * Đây là lớp cuối cùng của mạng, cung cấp kết quả đầu ra dựa trên các tính năng đã được trích xuất từ các lớp ẩn. Số lượng nút trong lớp này phụ thuộc vào số lượng lớp phân loại hoặc giá trị cần dự đoán.

Cách hoạt động của Weighted Sum và Activation Function

* **Weighted Sum**:
  + Mỗi neuron nhận đầu vào từ các neuron khác thông qua các synapse với trọng số tương ứng. Tổng hợp đầu vào được tính bằng cách nhân từng đầu vào với trọng số của nó và cộng lại:
  + Trong đó z*z* là tổng trọng số, wi*wi*​ là trọng số, xi*xi*​ là đầu vào và b*b* là độ lệch (bias).

**A diagram of a network

Description automatically generated**

**Activation Function**:

* Sau khi tính toán tổng trọng số, hàm kích hoạt sẽ quyết định xem neuron có nên "kích hoạt" hay không. Một số hàm kích hoạt phổ biến bao gồm:
* **ReLU (Rectified Linear Unit)**: f(z)=max⁡(0,z)*f*(*z*)=max(0,*z*)
* **Sigmoid**: f(z)=11+e−z*f*(*z*)=1+*e*−*z*1​
* **Tanh**: f(z)=tanh⁡(z)*f*(*z*)=tanh(*z*)

Hàm kích hoạt giúp thêm tính phi tuyến vào mô hình, cho phép DNN học hỏi từ dữ liệu phức tạp và tạo ra các dự đoán chính xác hơn.

1. **Các loại Neural Network phổ biến:**
   * **Fully-Connected Neural Network:** Đây là loại mạng đơn giản nhất, trong đó mỗi neuron của lớp này được kết nối với tất cả các neuron của lớp trước đó. Thông tin di chuyển một chiều từ lớp đầu vào đến lớp đầu ra. Mạng này thường được sử dụng cho các bài toán phân loại và hồi quy cơ bản.
   * **Convolutional Neural Network (CNN)**: CNN được thiết kế đặc biệt để xử lý dữ liệu dạng lưới, như hình ảnh. Nó sử dụng các lớp tích chập để trích xuất đặc trưng từ hình ảnh, giúp máy "nhìn" và "hiểu" thế giới xung quanh. CNN thường được ứng dụng trong nhận diện hình ảnh, phân loại hình ảnh và nhiều tác vụ liên quan đến thị giác máy tính.
   * **Recurrent Neural Network (RNN)**: RNN là mạng nơ-ron có khả năng xử lý dữ liệu chuỗi, cho phép lưu trữ thông tin từ các bước xử lý trước đó. Điều này giúp RNN học hỏi mối quan hệ phụ thuộc trong dữ liệu theo thời gian, rất hữu ích cho các bài toán như phân tích ngôn ngữ tự nhiên và dự đoán chuỗi thời gian.
   * **Long Short-Term Memory (LSTM)**: LSTM là một biến thể của RNN, được thiết kế để giải quyết vấn đề vanishing gradient trong quá trình huấn luyện. LSTM có khả năng ghi nhớ thông tin lâu dài và ngắn hạn, rất hiệu quả trong các tác vụ như dịch máy và nhận diện giọng nói,
   * **Transformers**: Transformers là kiến trúc mạng nơ-ron hiện đại, nổi bật trong việc xử lý ngôn ngữ tự nhiên. Khác với RNN, Transformers không dựa vào tuần tự mà sử dụng cơ chế attention để xử lý dữ liệu song song, giúp cải thiện tốc độ và hiệu suất trong các tác vụ như dịch thuật và phân tích văn bản.
   * A diagram of a network

     Description automatically generated

A diagram of a neural network

Description automatically generated

**III. Kiến trúc CNN chi tiết:**

1. **Convolutional Layer (Lớp Tích Chập)**:
   * Đây là lớp quan trọng nhất trong CNN, thực hiện phép toán tích chập giữa đầu vào và các bộ lọc (filters) để trích xuất đặc trưng từ dữ liệu.
   * Mỗi bộ lọc di chuyển qua hình ảnh, tạo ra các feature map, thể hiện các đặc trưng đã được phát hiện.
2. **Pooling Layer (Lớp Gộp)**:
   * Lớp này giúp giảm kích thước của feature map từ lớp convolutional, giúp giảm độ phức tạp tính toán và ngăn chặn overfitting.
   * Các phương pháp pooling phổ biến bao gồm max pooling (lấy giá trị lớn nhất) và average pooling (lấy giá trị trung bình).
3. **Normalization Layer (Lớp Chuẩn Hóa)**:
   * Lớp này thường được sử dụng để chuẩn hóa đầu ra của các lớp trước đó, giúp cải thiện tốc độ hội tụ và hiệu suất của mạng.
   * Một ví dụ phổ biến là Batch Normalization, giúp điều chỉnh phân phối đầu ra của mỗi lớp.
4. **Fully Connected Layer (Lớp Kết Nối Đầy Đủ)**:
   * Đây là lớp cuối cùng trong CNN, nơi mà đầu ra từ các lớp trước được kết nối đầy đủ để thực hiện phân loại hoặc hồi quy.
   * Mỗi neuron trong lớp này kết nối với tất cả các neuron của lớp trước đó.

Khái niệm về Filter, Feature Map và Cách Chúng Hoạt Động

* **Filter (Bộ Lọc)**: Là ma trận nhỏ (thường có kích thước 3x3 hoặc 5x5) chứa các trọng số, dùng để quét qua ảnh đầu vào.
* **Feature Map**: Là kết quả đầu ra từ phép tích chập, thể hiện các đặc trưng mà bộ lọc đã phát hiện. Mỗi bộ lọc sẽ tạo ra một feature map riêng.
* **Cách Hoạt Động**: Khi bộ lọc di chuyển qua ảnh, nó sẽ phát hiện các đặc trưng như cạnh, góc hoặc hình dạng cụ thể. Nhiều bộ lọc khác nhau sẽ giúp mạng học được nhiều đặc trưng khác nhau từ hình ảnh.

Tầm Quan Trọng của Stride và Padding

* **Stride**: Là bước di chuyển của bộ lọc khi quét qua ảnh. Stride lớn hơn 1 sẽ giảm kích thước của feature map, trong khi stride bằng 1 giữ nguyên kích thước.
* **Padding**: Là việc thêm các giá trị 0 xung quanh ảnh đầu vào để giữ nguyên kích thước sau khi thực hiện phép tích chập. Padding giúp bảo toàn thông tin ở biên của ảnh và cải thiện khả năng học hỏi của mạng.

Input/Output Channels

* **Input Channels**: Là số lượng kênh đầu vào của dữ liệu. Ví dụ, một bức ảnh RGB có 3 kênh (đỏ, xanh lá cây, xanh dương).
* **Output Channels**: Là số lượng feature maps mà convolutional layer tạo ra. Mỗi bộ lọc sẽ tạo ra một output channel tương ứng với một feature map.

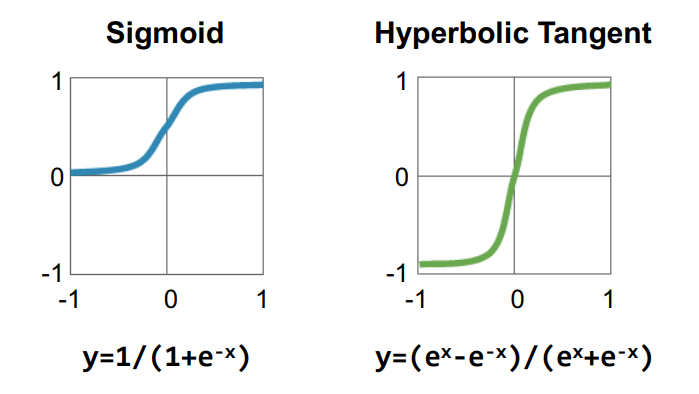
**IV. Activation Functions:**

1. **Sigmoid**

* **Ưu điểm**:
  + Đầu ra nằm trong khoảng (0, 1), dễ dàng sử dụng cho các bài toán phân loại nhị phân.
  + Dễ hiểu và dễ triển khai.
* **Nhược điểm**:
  + Vấn đề vanishing gradient: Gradient gần bằng 0 khi đầu vào lớn hoặc nhỏ, làm cho việc huấn luyện mạng trở nên chậm chạp.
  + Không trung tâm tại 0, có thể dẫn đến việc không tối ưu hóa hiệu quả.

1. **Hyperbolic Tangent (tanh)**

* **Ưu điểm**:
  + Đầu ra nằm trong khoảng (-1, 1), giúp cải thiện tốc độ hội tụ hơn so với sigmoid.
  + Trung tâm tại 0, giúp giảm thiểu bias trong quá trình huấn luyện.
* **Nhược điểm**:
  + Cũng gặp vấn đề vanishing gradient tương tự như sigmoid.



2. **Activation Functions Hiện Đại**

* **ReLU (Rectified Linear Unit)**
* **Ưu điểm**:
  + Giảm thiểu vấn đề vanishing gradient; gradient luôn bằng 1 cho các giá trị dương.
  + Tính toán đơn giản và nhanh chóng.
* **Nhược điểm**:
  + Vấn đề "dying ReLU": Neuron có thể trở nên không hoạt động nếu đầu vào luôn âm.
* **Leaky ReLU**
* **Ưu điểm**:
  + Giải quyết vấn đề dying ReLU bằng cách cho phép một lượng nhỏ gradient khi đầu vào âm.
* **Nhược điểm**:
  + Gradient vẫn có thể nhỏ đối với các giá trị âm, có thể ảnh hưởng đến khả năng học.
* **ELU (Exponential Linear Unit)**
* **Ưu điểm**:
  + Giảm thiểu vấn đề vanishing gradient và dying ReLU.
  + Đầu ra âm giúp trung tâm tại 0, cải thiện tốc độ hội tụ.
* **Nhược điểm**:
  + Tính toán phức tạp hơn so với ReLU.
* **Swish**
* **Ưu điểm**:
  + Hiệu suất tốt hơn trên nhiều bài toán so với ReLU và tanh.
  + Không gặp vấn đề dying neuron.
* **Nhược điểm**:
  + Tính toán phức tạp hơn, làm cho việc triển khai chậm hơn một chút.

A graph of a function

Description automatically generated **ReLU đạt tỷ lệ lỗi huấn luyện 25% trên CIFAR-10 nhanh gấp sáu lần so với tanh.**



A table with numbers and letters

Description automatically generated Kết luận

* **Sigmoid và Tanh**: Thường được sử dụng trong các mạng nhỏ hoặc bài toán cụ thể như phân loại nhị phân (Sigmoid) hoặc xử lý dữ liệu chuỗi (Tanh). Tuy nhiên, chúng gặp vấn đề với vanishing gradient trong các mạng sâu.
* **ReLU**: Là lựa chọn mặc định cho các mạng nơ-ron hiện đại do tính hiệu quả và khả năng giảm thiểu vấn đề vanishing gradient. Tuy nhiên, cần cẩn thận với vấn đề "dying ReLU" và có thể sử dụng các biến thể như Leaky ReLU hoặc ELU để khắc phục.

**V.** **Pooling và Normalization:**

* 1. So sánh Max Pooling và Average Pooling:



* 1. Batch Normalization và Tác Động của Nó:
* **Khái niệm**: Batch Normalization là một kỹ thuật chuẩn hóa đầu ra từ các lớp trước đó trong mạng nơ-ron, giúp cải thiện tốc độ hội tụ và hiệu suất của mô hình.
* **Cách hoạt động**:
  1. Chuẩn hóa đầu ra của mỗi mini-batch bằng cách tính toán giá trị trung bình và phương sai.
  2. Cung cấp tham số điều chỉnh (gamma và beta) để điều chỉnh lại đầu ra sau khi chuẩn hóa.
* **Tác động**:
  1. Giảm thiểu vấn đề vanishing gradient, giúp mạng học nhanh hơn.
  2. Cải thiện độ chính xác của mô hình.
  3. Giúp giảm thiểu overfitting nhờ vào việc tạo ra một dạng regularization

3. Normalization-Free Networks

* **Khái niệm**: Đây là các mạng nơ-ron không sử dụng bất kỳ kỹ thuật chuẩn hóa nào như Batch Normalization.
* **Lợi ích**:
  + Đơn giản hóa kiến trúc mạng, giảm bớt số lượng tham số cần điều chỉnh.
  + Có thể hoạt động tốt trong một số trường hợp nhất định, đặc biệt là với các kiến trúc mạng hiện đại như ResNet, nơi mà việc sử dụng shortcut connections giúp duy trì gradient hiệu quả mà không cần chuẩn hóa.
* **Ví dụ**: Một số mô hình mới như EfficientNet đã chứng minh rằng có thể đạt được hiệu suất cao mà không cần sử dụng Batch Normalization

**VI. Quy trình huấn luyện và suy luận**

1. Forward Propagation

* Khái niệm: Forward propagation là quá trình tính toán đầu ra của mạng nơ-ron từ đầu vào thông qua các lớp khác nhau.
* Quy trình:
  1. Nhận đầu vào: Dữ liệu đầu vào được cung cấp cho lớp đầu tiên của mạng.
  2. Tính toán tại mỗi lớp:
     + Đầu vào được nhân với trọng số (weights) và cộng với độ lệch (bias).
     + Kết quả này sau đó được đưa qua hàm kích hoạt (activation function) để tạo ra đầu ra của lớp.
  3. Tiến đến lớp tiếp theo: Kết quả từ lớp trước sẽ trở thành đầu vào cho lớp tiếp theo, và quá trình này tiếp tục cho đến khi đạt đến lớp đầu ra.
  4. Kết quả cuối cùng: Đầu ra từ lớp cuối cùng là dự đoán của mạng nơ-ron.

2. Back Propagation

* **Khái niệm**: Back propagation là quá trình điều chỉnh trọng số của mạng nơ-ron dựa trên lỗi giữa đầu ra dự đoán và đầu ra thực tế.
* **Quy trình**:
  1. **Tính toán lỗi**: Sử dụng hàm mất mát (loss function) để tính toán sai lệch giữa đầu ra dự đoán và giá trị thực tế.
  2. **Lan truyền ngược**:
     + Lỗi được lan truyền ngược qua mạng từ lớp đầu ra về lớp đầu vào.
     + Tại mỗi lớp, gradient của hàm mất mát đối với trọng số được tính toán bằng cách sử dụng quy tắc chuỗi (chain rule).
  3. **Cập nhật trọng số**: Trọng số được cập nhật bằng cách trừ đi gradient đã tính nhân với tốc độ học (learning rate).

3. Loss Function

* **Khái niệm**: Loss function (hàm mất mát) đo lường mức độ sai lệch giữa giá trị dự đoán của mô hình và giá trị thực tế.
* **Các loại hàm mất mát phổ biến**:
  + **Mean Squared Error (MSE)**:
    - Sử dụng cho các bài toán hồi quy.
  + **Cross-Entropy Loss**:
* Thường dùng cho các bài toán phân loại nhị phân hoặc đa lớp.

4. Gradient Descent

* **Khái niệm**: Gradient descent là thuật toán tối ưu hóa được sử dụng để cập nhật trọng số của mạng nơ-ron nhằm giảm thiểu hàm mất mát.

A diagram of a training

Description automatically generated

================================================================

Chương 3: Mô hình Deep Neural Network phổ biến

A. Các mô hình CNN nổi bật

* **LeNet (1998)**
  + **Cấu trúc**: 2 lớp convolution, 2 lớp fully connected.
  + **Số lượng trọng số**: 60k.
  + **MACs**: 341k.
  + **Ứng dụng**: Nhận diện chữ số trong dataset MNIST.

A diagram of a diagram of a diagram

Description automatically generated with medium confidence

* **AlexNet (2012)**
  + **Cấu trúc**: 5 lớp convolution, 3 lớp fully connected.
  + **Số lượng trọng số**: 61M.
  + **MACs**: 724M.
  + **Đặc điểm**: Sử dụng ReLU, Local Response Normalization (LRN).
  + **Thành tựu**: Giành chiến thắng tại ILSVRC 2012.

A diagram of a block diagram

Description automatically generated

* **VGGNet-16**
  + **Cấu trúc**: 13 lớp convolution, 3 lớp fully connected.
  + **Số lượng trọng số**: 138M.
  + **MACs**: 15.5G.
  + **Đặc điểm**: Sử dụng stacked filters với kích thước nhỏ (3x3).
  + **Stacked filters:** Kỹ thuật này sử dụng nhiều bộ lọc nhỏ (thường là 3x3) thay vì một bộ lọc lớn (như 5x5 hoặc 7x7) để giảm số lượng trọng số trong mạng nơ-ron mà vẫn giữ được khả năng nhận diện các đặc trưng phức tạp.
  + Trong VGG-16, kiến trúc mạng bao gồm các lớp convolution xếp chồng lên nhau với kích thước 3x3. Mỗi nhóm hai lớp convolution sẽ được theo sau bởi một lớp pooling để giảm kích thước đầu ra và tăng cường khả năng tổng quát của mô hình.

A diagram of a graph

Description automatically generated with medium confidence A grid with numbers and arrows

Description automatically generated

* **GoogleNet/Inception (2014)**
* **Cấu trúc**:
  + Gồm nhiều Inception blocks, tổng cộng có 21 lớp convolution và một lớp fully connected.
* **Số lượng trọng số**: Khoảng 7 triệu.
* **MACs**: ~1.43 tỷ.
* **Đặc điểm**:
  + Sử dụng các bộ lọc song song với kích thước khác nhau để xử lý dữ liệu ở nhiều tỷ lệ khác nhau.
  + Giảm số lượng trọng số bằng cách sử dụng global pooling.

A diagram of a number of classes

Description automatically generated A diagram of a flowchart

Description automatically generated

**Bottleneck** trong Deep Neural Networks (DNNs) là một thuật ngữ chỉ một cấu trúc hoặc giai đoạn trong mạng nơ-ron nơi mà số lượng kênh (channels) hoặc số lượng trọng số (weights) được giảm xuống một cách có chủ đích để tối ưu hóa hiệu suất và giảm độ phức tạp tính toán. Cấu trúc bottleneck thường được sử dụng để cải thiện hiệu quả của mạng mà không làm giảm chất lượng đầu ra.

ResNet (2015)

* **Cấu trúc**:
  + Gồm nhiều phiên bản với độ sâu khác nhau (ResNet-50, ResNet-152).
* **Số lượng trọng số**: Khoảng từ 25.5 triệu đến hơn 60 triệu tùy phiên bản.
* **MACs**: ~3.86 tỷ cho ResNet-50.
* **Đặc điểm**:
  + Sử dụng shortcut connections để giải quyết vấn đề vanishing gradient, cho phép xây dựng mạng nơ-ron rất sâu mà không gặp phải vấn đề này.

**A diagram of a complex algorithm

Description automatically generated with medium confidence**

**A diagram of a compress

Description automatically generated with medium confidence**

**DenseNet**

* **Cấu trúc:**
  + Kết nối dày đặc giữa các lớp, cho phép tái sử dụng feature maps và cải thiện độ chính xác với ít trọng số hơn.

A diagram of a graph

Description automatically generated

1. Convolution

* Mô hình hóa phụ thuộc: Convolution chỉ mô hình hóa các phụ thuộc giữa các điểm lân cận trong không gian (spatial neighbors). Điều này có nghĩa là nó chỉ xem xét các pixel gần nhau trong một hình ảnh mà không tính đến các mối quan hệ xa hơn.
* Kết nối thưa: Các lớp convolution sử dụng kết nối thưa với các điểm lân cận, dẫn đến việc không hỗ trợ cho các phụ thuộc bên ngoài các chiều không gian của bộ lọc (R x S). Điều này có thể hạn chế khả năng của mạng trong việc học các đặc trưng toàn cục từ dữ liệu.

2. Attention

* Mô hình hóa phụ thuộc toàn cục: Kỹ thuật attention cho phép mô hình hóa các phụ thuộc toàn cục mà không cần quan tâm đến khoảng cách giữa chúng. Điều này có nghĩa là nó có thể chú ý đến bất kỳ phần nào của đầu vào, bất kể vị trí của nó.
* Chi phí tính toán: Tuy nhiên, việc sử dụng một lớp fully connected để thực hiện attention có thể rất tốn kém về mặt tính toán. Do đó, cần phát triển cơ chế để thiên về "phân bổ tài nguyên tính toán có sẵn cho các thành phần thông tin nhất của tín hiệu". Điều này giúp tối ưu hóa việc sử dụng tài nguyên và cải thiện hiệu suất.

3. Transformer

* Kiến trúc DNN: Transformer là một loại mạng nơ-ron sâu hoàn toàn được xây dựng dựa trên cơ chế attention. Kiến trúc này đã được giới thiệu trong bài báo "Attention is All You Need" bởi Vaswani et al. tại NeurIPS 2017 và đã trở thành nền tảng cho nhiều ứng dụng trong xử lý ngôn ngữ tự nhiên và nhiều lĩnh vực khác.

B. Giảm số kênh (C)

1. **1x1 Convolution**:
   * Sử dụng bộ lọc 1x1 để giảm số lượng kênh mà không làm mất thông tin không gian.
   * Kỹ thuật này cho phép tăng cường khả năng học hỏi và cải thiện hiệu suất của mạng.
2. **Grouped Convolution**:
   * Chia các kênh đầu vào thành nhiều nhóm và thực hiện convolution riêng cho từng nhóm.
   * Điều này giúp giảm thiểu số lượng phép toán và trọng số trong mạng.
3. **Depth-wise Convolution**:
   * Thực hiện convolution riêng cho từng kênh đầu vào, giúp giảm thiểu số lượng phép toán và trọng số.
   * Kỹ thuật này thường được sử dụng trong các mô hình nhẹ như MobileNets.

C. Giảm số bộ lọc (M)

1. **Feature Map Reuse**:
   * Tái sử dụng các feature map giữa các lớp để giảm thiểu số lượng bộ lọc cần thiết.
   * Điều này giúp tối ưu hóa tài nguyên tính toán mà vẫn duy trì hiệu suất của mạng.
2. **Dense Connections**:
   * Kết nối nhiều lớp lại với nhau, cho phép mỗi lớp truy cập vào tất cả các feature map từ các lớp trước đó.
   * Kỹ thuật này giúp cải thiện khả năng tái sử dụng thông tin và tăng cường độ chính xác của mô hình.

Neural Architecture Search (NAS)

A. Các thành phần chính

Neural Architecture Search (NAS) là một quy trình tự động hóa để tìm kiếm và tối ưu hóa kiến trúc mạng nơ-ron. Các thành phần chính của NAS bao gồm:

1. **Search Space**:
   * Đây là tập hợp các kiến trúc mạng có thể được khám phá trong quá trình tìm kiếm. Search space xác định các loại lớp, số lượng lớp, cách kết nối giữa các lớp, và các tham số khác có thể thay đổi để tạo ra các mô hình khác nhau.
2. **Optimization Algorithm**:
   * Thuật toán tối ưu hóa được sử dụng để tìm kiếm trong không gian kiến trúc nhằm tối ưu hóa hiệu suất của mạng. Các thuật toán này có thể bao gồm các phương pháp như reinforcement learning, evolutionary algorithms, hoặc gradient-based methods để tìm kiếm kiến trúc tốt nhất.
3. **Performance Evaluation**:
   * Đánh giá hiệu suất của các kiến trúc được khám phá thông qua các chỉ tiêu như độ chính xác, tốc độ huấn luyện và số lượng tham số. Điều này thường được thực hiện bằng cách huấn luyện và kiểm tra mô hình trong một tập dữ liệu cụ thể để đảm bảo rằng nó hoạt động tốt trong thực tế.D

A diagram of a graph

Description automatically generated

1. NASNet

* **Khái niệm**: NASNet là một kiến trúc mạng nơ-ron được tự động thiết kế thông qua quy trình Neural Architecture Search. Nó được phát triển bởi Google Brain và đã giành được giải thưởng trong các cuộc thi phân loại hình ảnh.
* **Đặc điểm**:
  + Sử dụng các kiến trúc con (cell) được tối ưu hóa để tạo thành một mạng nơ-ron lớn hơn.
  + Thực hiện tốt trên các bài toán phân loại hình ảnh với độ chính xác cao, nhờ vào việc tự động hóa quy trình thiết kế.
* **Hiệu suất**: NASNet đã đạt được kết quả tốt trong các thử nghiệm trên tập dữ liệu ImageNet, cho thấy khả năng của NAS trong việc tạo ra các kiến trúc hiệu quả mà không cần can thiệp thủ công.

2. EfficientNet

* **Khái niệm**: EfficientNet là một dòng kiến trúc mạng nơ-ron được phát triển bằng cách sử dụng NAS để tối ưu hóa kích thước và hiệu suất của mạng.
* **Đặc điểm**:
  + Kết hợp ba yếu tố: chiều sâu, chiều rộng và độ phân giải của mạng để tối ưu hóa hiệu suất mà không làm tăng quá nhiều số lượng tham số.
  + Sử dụng kỹ thuật compound scaling để điều chỉnh các yếu tố này đồng thời, giúp đạt được sự cân bằng tốt nhất giữa độ chính xác và tài nguyên tính toán.
* **Hiệu suất**: EfficientNet đã đạt được độ chính xác cao trên nhiều bài toán phân loại hình ảnh, đồng thời giảm thiểu số lượng trọng số và phép toán cần thiết so với các kiến trúc trước đó.

A graph of numbers and a line

Description automatically generated

D. So sánh và đánh giá giữa các mô hình:

A black and white screen with white text

Description automatically generated