# Xử Lý Bài Toán Sudoku bằng CNN

1st Trương Thị Cẩm Ly
*Khoa Công Nghệ Thông Tin*
*Chuyên ngành Khoa Học Dữ Liệu*
truongthicamly1412@gmail.com
MSSV: 19532211

2nd Quách Trọng Nghĩa
*Khoa Công nghệ thông tin*
*Chuyên ngành Khoa Học Dữ Liệu*
26quachtrongnghia@gmail.com
MSSV: 19502841

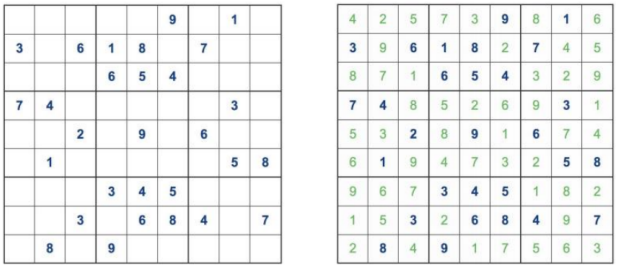
**TÓM TẮT NỘI DUNG**

Sudoku là một trong những trò chơi dựa trên logic phổ biến nhất mọi thời đại, giúp phát triển khả năng suy luận quan hệ và phát triển mô hình của các cá nhân. Chúng tôi sử dụng Convolution Neural Networks(CNN) để giải các câu đố. Mô hình CNN hiệu quả về mặt tính toán, cho phép đào tạo tốt, do đó mang lại độ chính xác của training và test accuracies cao.

## I. GIỚI THIỆU NỘI DUNG

Các trò chơi dựa trên logic như Sudoku đã được chứng minh là có thể giúp trì hoãn các chứng rối loạn thần kinh như Alzheimer và sa sút trí tuệ. Khi giải các câu đố Sudoku, người chơi phải dựa vào vị trí của các con số trên bàn cờ và phụ thuộc vào thời gian để hoàn thành câu đố.

Mục tiêu của trò chơi là điền các chữ số vào một lưới 9×9 sao cho mỗi cột, mỗi hàng, và mỗi phần trong số chín lưới con 3×3 cấu tạo nên lưới chính (cũng gọi là "hộp", "khối", hoặc "vùng") đều chứa tất cả các chữ số từ 1 tới 9. Câu đố đã được hoàn thành một phần, người chơi phải giải tiếp bằng việc điền số. Mỗi câu đố được thiết lập tốt có một cách làm duy nhất. Được thúc đẩy bởi những ngữ nghĩa này, chúng tôi quyết định giải các câu đố Sudoku bằng CNN. Phương pháp này sử dụng suy luận quan hệ, đòi hỏi các mạng phải rút ra những kết luận 'logic' về mối quan hệ giữa các con số trong câu đố. Đầu vào cho các mạng là mảng có độ dài 81 phần tử đại diện cho số trên bảng sudoku 9x9 chưa được giải quyết. Kết quả đầu ra là vector 81 phần tử đại diện cho độ dài của output của các bảng sudoku đã giải, như mô tả ở Hình 1.



Hình 1: Bảng sudoku mẫu hiển thị câu hỏi (đầu vào) và giải pháp của nó (đầu ra mong muốn)

## II. NGHIÊN CỨU LIÊN QUAN

**Tìm kiếm thuật toán**: Cách tiếp cận phổ biến nhất cho việc xử lý sudoku là thuật toán tree search, heuristic search và quay lui. Những phương pháp này cho phép kết hợp các quy tắc của sudoku vào quá trình giải quyết vấn đề, làm tăng khả năng hội tụ về một giải pháp. Nhưng các phương pháp này sẽ đi kèm với sự phức tạp của thuật toán, tốn rất nhiều thời gian để có thể giải hết được cả một bộ dữ liệu phức tạp [1] [2]. Với lý do này, để giải được những câu đố sudoku phức tạp này, chúng ta nên sử dụng phương pháp tiếp cận máy học, cụ thể hơn là sử dụng Convolution Neural Networks để giải quyết.

Ngoài Convolution Newral Networks thì cũng có thể giải quyết bài toán Sudoku bằng LSTM hay Seq2Seq. Nhưng mang lại kết quả chính xác không cao.

OptNet: Differentiable Optimization as a Layer in Neural Networks giải quyết sudoku bằng cách sử dụng kiến trúc mạng tích hợp tối ưu hóa dưới dạng các bài toán bậc hai. Việc này cho phép mạng tìm hiểu các vấn đề về sudoku hoàn toàn từ dữ liệu với độ chính xác cao. Hạn chế chính của cách tiếp cận này là nó có độ phức tạp lớn đối với số lượng biến và / hoặc các ràng buộc có nghĩa là số lượng hidden layer phải nhỏ để đảm bảo việc dự đoán mang lại độ chính xác cao hơn [3].

## III. BỘ DỮ LIỆU

Bộ dữ liệu được sử dụng trong bài báo này là 1 Million Sudoku games được lấy từ Kaggle của tác giả Kyubyong Park. Để xây dựng câu đố hợp lý, thường để lại 46-49 ô trống trong bảng, kết quả câu đố được dùng làm đầu vào và ghép nối với các giải pháp tương quan, mỗi cặp (quiz, solution) được sử dụng làm input đầu vào. Mỗi quiz hay mỗi Solution đều là một chuỗi gồm 81 ký tự chứa các số từ 0-9 tương ứng với các số có trong bảng sudoku và lưu vào tệp csv như hình 2

| | |
|---|---|
| Quiz | 000009010306180700000654000740000<br>030002090600010000058000345000003<br>068407080900000 |
| Solution | 425739816396182745871654329748526 9<br>315328916746194732589673451821532 68<br>497284917563 |

Hình 2: Dataset Examples

Từ một chuỗi ký tự gồm 81 phần tử sau đó được tách ra thành mảng 2 chiều gồm: 9x9 phần tử và lưu vào tệp csv. Từ 1.000.000 dòng dữ liệu ban đầu, sử dụng 800.000 mẫu được lưu giữ cho tập training set và 200.000 mẫu được lưu giữ cho tập testing set.

## IV. PHƯƠNG PHÁP TIẾP CẬN

Kiến trúc mạng học sâu đã được nhóm áp dụng trong bài toán này là: Convolution Neural NetWorks. Phương pháp này sử dụng hàm kích hoạt relu ở mỗi lớp tích chập và sử dụng hàm kích hoạt softmax ở lớp cuối cùng để tạo ra giá trị xác suất trên K nhãn mong đợi: K = (1, 2, ..., 9). Sử dụng:

$$P(C_k|x) = y_k(x) = \frac{e^{a_k}}{\sum_{j=1}^{K} e^{a_j}} \quad (1)$$

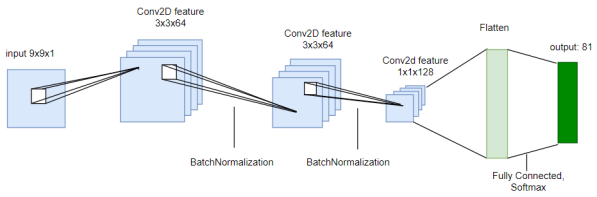
Trong đó x đại diện cho kết quả đầu ra từ lớp trước,và $a_k = W_k^T x + b_k$, với $W_k$ và $b_k$ đại diện cho trọng số của lớp softmax và bias của lớp K. Sau đó, thực hiện lấy argmax trên các xác suất để chọn label có khả năng xảy ra cao nhất cho một ô nhất định trong bảng sudoku. Hàm loss được sử dụng trong mô hình là Sparse Multiclass Cross-Entropy Loss:

$$\text{Loss} = -\sum_{i=1}^{K} y_i \cdot \log \hat{y}_i \quad (2)$$

Cuối cùng, mô hình sử dụng hàm tối ưu Adam, với learning rate là: $\theta = 0.001$

### A. Baseline Model

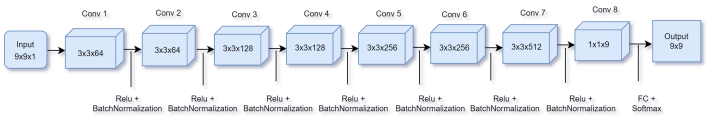
Xây dựng một mô hình baseline đơn giản nhưng qua đó có thể có được cái nhìn sâu hơn về bài toán để xây dựng một mô hình tốt hơn. Kiến trúc của mô hình baseline được mô tả ở Hình 3. Mô hình này sử dụng 3 lớp Conv2D.




Hình 3: Kiến trúc mô hình Baseline

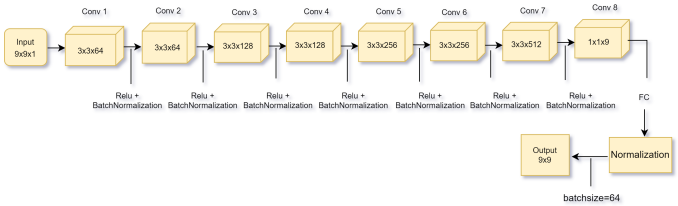
### B. Sử dụng thêm lớp Convolution

Dựa trên mô hình baseline ở trên, nhóm đã quyết định thêm một số lớp Convolution vào để mô hình học được nhiều ngữ cảnh hơn. Cụ thể là thêm 5 lớp Convolution, sử dụng BathNormalization và hàm kích hoạt relu ở sau mỗi lớp. Sau khi thử nghiệm với trường hợp này, mô hình có sự cải tiến đáng kể, việc dự đoán kết quả cũng khá tốt. Kiến trúc mô hình được mô tả cụ thể như Hình 4. Từ kiến trúc mô hình này, để tăng độ chính xác của mô hình, nhóm đã tiếp tục làm một số thử nghiệm khác.




Hình 4: Kiến trúc mô hình 8 lớp Conv với batchsize = 32

Nhóm đã tiếp tục cải thiện mô hình trên bằng cách tăng dữ liệu huấn luyện lên 800.000, sử dụng batch size là 64 và thêm một lớp LayerNorm vào sau lớp Fully Connected để cải tiến tốc đố huấn luyện với các mô hình neural nerworks (Hình 5). Không giống như batch normalization, phương pháp này ước tính trực tiếp số liệu thống kê chuẩn hóa từ các đầu vào tổng hợp đến các nơ-ron bên trong một lớp hidden layer. Thêm vào đó, để tránh việc overfiting, mô hình này được sử dụng thêm EarlyStopping, nó sẽ ngừng đào tạo mô hình khi chỉ số của mô hình ngừng cải thiện. Kiến trúc mô hình này được mô tả cụ thể như sau:




Hình 5: Kiến trúc mô hình 8 lớp Conv với batchsize = 64 và một lớp Normalization

## V. KẾT QUẢ THỰC NGHIỆM

### A. *Chỉ số đánh giá*

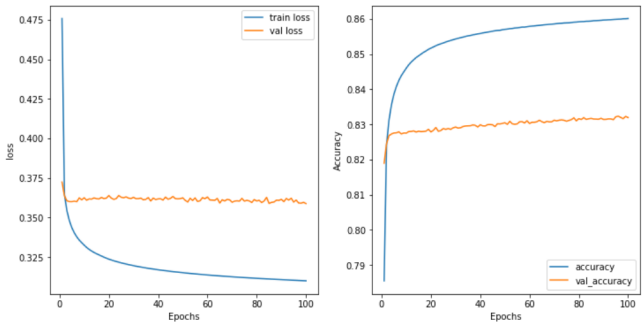
Các chỉ số chính được sử dụng để đánh giá mô hình là Accuracy và Loss. Accuracy tính toán tỷ lệ phần trăm các ô được gán nhãn chính xác qua một tập hợp các câu đố, accuracy được tính bằng cách tính trung bình độ chính các của từng cell trong mỗi một câu đố khác nhau. Accuracy càng tiến về 1 thì độ chính xác của mô hình càng cao. Công thức tính Accuracy:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

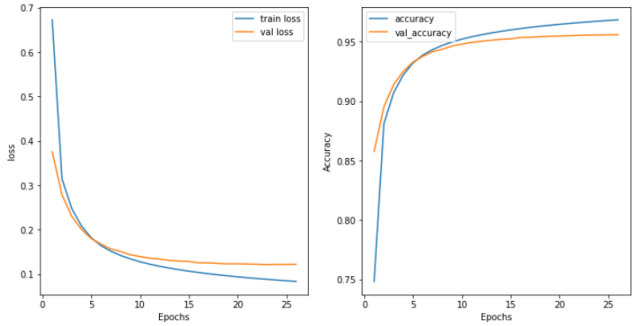
Ngược lại với Accuracy, Loss sẽ chỉ ra mô hình dự đoán sai bao nhiêu % (tính theo cell) so với thực tế. Do vậy, loss càng tiến về 0 thì mô hình có độ chính xác càng cao.(Nhóm sử dụng hàm Sparse Multiclass Cross-Entropy Loss được mô tả ở công thức 2 ).

### B. *Kết quả*

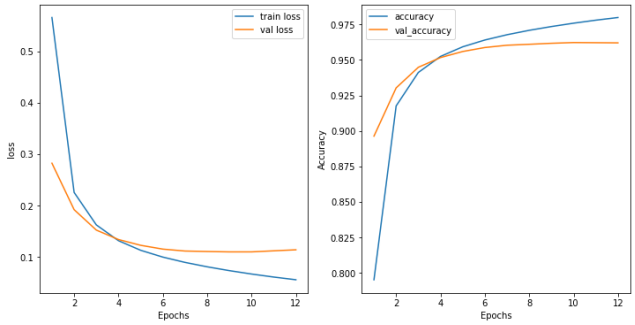
Ở mô hình baseline, độ chính xác của mô hình tương đối thấp, cụ thể là 83%. Ở mô hình 8 lớp Convolution, $batchsize = 32$, kết quả mô hình cải thiện lên đến 96 %. Với mô hình đã thêm LayerNormalization độ chính xác đã được nâng lên 98%. Độ chính xác và sai số của các mô hình được trực quan hóa qua các biểu đồ sau:




Hình 6: Độ chính xác và sai số của mô hình baseline




Hình 7: Độ chính xác và sai số của mô hình 8 lớp Conv và batchsize=32




Hình 8: Độ chính xác và sai số của mô hình 8 lớp Conv và batchsize=64

Dựa vào kết quả trên, nhóm đã chọn mô hình cuối làm mô hình chính để xử lý các bài toán sudoku. Đây là kết quả output của mô hình 8 lớp CNN kết hợp với LayerNormalization:

```
inp
```

```
array([[0, 4, 6, 5, 0, 0, 1, 0, 0],
       [7, 0, 2, 3, 0, 0, 0, 0, 5],
       [0, 1, 0, 0, 8, 0, 2, 0, 0],
       [0, 8, 0, 0, 3, 0, 7, 4, 0],
       [3, 0, 9, 1, 6, 0, 0, 0, 0],
       [0, 5, 0, 0, 7, 0, 0, 6, 0],
       [0, 0, 0, 9, 2, 0, 8, 0, 7],
       [0, 3, 0, 0, 0, 6, 0, 5, 2],
       [1, 0, 0, 4, 0, 8, 9, 0, 0]])
```

Hình 9: bảng sudoku đầu vào cần giải

```
y_
```

```
array([[8, 4, 6, 5, 9, 2, 1, 7, 3],
       [7, 9, 2, 3, 4, 1, 6, 8, 5],
       [5, 1, 3, 6, 8, 7, 2, 9, 4],
       [6, 8, 1, 2, 3, 5, 7, 4, 9],
       [3, 7, 9, 1, 6, 4, 5, 2, 8],
       [2, 5, 4, 8, 7, 9, 3, 6, 1],
       [4, 6, 5, 9, 2, 3, 8, 1, 7],
       [9, 3, 8, 7, 1, 6, 4, 5, 2],
       [1, 2, 7, 4, 5, 8, 9, 3, 6]])
```

(a) Đầu ra mong muốn

```
model.predict(inp.reshape(1, 9,
```

```
array([[8, 4, 6, 5, 9, 2, 1, 7, 3],
       [7, 9, 2, 3, 4, 1, 6, 8, 5],
       [5, 1, 3, 6, 8, 7, 2, 9, 4],
       [6, 8, 1, 2, 3, 5, 7, 4, 9],
       [3, 7, 9, 1, 6, 4, 5, 2, 8],
       [2, 5, 4, 8, 7, 9, 3, 6, 1],
       [4, 6, 5, 9, 2, 3, 8, 1, 7],
       [9, 3, 8, 7, 1, 6, 4, 5, 2],
       [1, 2, 7, 4, 5, 8, 9, 3, 6]])
```

(b) Đầu ra của mô hình

Hình 10: Kết quả mô hình

### C. *So sánh mô hình*

Dựa vào kết quả thử nghiệm thu được (Hình 11), mô hình với 8 lớp Convolution kết hợp với LayerNormalization đưa ra được kết quả đáng mong đợi nhất, nhưng lại mất khá nhiều thời gian. Cụ thể việc so sánh 3 mô hình này được mô tả ở Hình 11

| Architecture | Epoch | Time | Train Accuracy | Test Accuracy | Train Loss | Test Lost |
|---|---|---|---|---|---|---|
| 3 Layer CNN, batchsize=32 | 100 | 5h | 0.8503 | 0.8364 | 0.3315 | 0.8364 |
| 8 Layer CNN, batchsize=32 | 26 | 6h | 0.9685 | 0.9559 | 0.098 | 0.1213 |
| 8 Layer CNN, Batchsize=64 + LayerNormalization | 12 | 5h | 0.98 | 0.962 | 0.0555 | 0.1138 |

Hình 11: Độ chính xác và sai số của các mô hình

## VI. HƯỚNG PHÁT TRIỂN

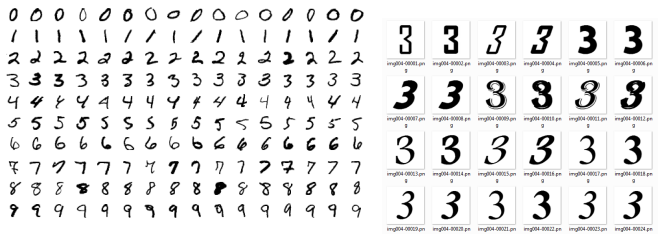
SudoKu là một trò Logic rất phổ biến, đòi hỏi tính suy luận cao. Vì vậy việc giải Sudoku cũng gặp nhiều khó khăn đối với những người mới chơi hoặc đối với các chuyên gia khi giải những bài toán Sudoku khó. Vì vậy, nhóm đã có hướng phát triển mới là sẽ đọc một ảnh chụp hình Sudoku bất kì mà người chơi muốn giải. Sau đó Detect các số trong hình đó ra một ma trận, đối với những ô bị trống thì mặc định gán ô đó với giá trị bằng 0 và đưa vào mô hình CNN ở trên để cho ra giải pháp của bài toán đó. Chúng tôi đã lấy tập dữ liệu ảnh Sudoku trên Kaggle, với 2100 ảnh để test việc detect ảnh đúng hay không.

Việc detect các số trong ảnh ra phải trải qua một số giai đoạn. Giai đoạn đầu tiên, chúng tôi đã huấn luyện một mô hình phân loại chữ số. Dữ liệu chúng tôi dùng là gộp từ 2 tập dữ liệu: MNIST và Chars74K, bao gồm cả chữ số viết tay và chữ số in như Hình 12. Việc huấn luyện mô hình này sẽ mang lại kết quả tốt khi chúng ta cần nhận diện các chữ số trong hình sudoku là số nào để đưa nó vào mô hình giải câu đố.

Phương pháp tiếp cận trong bài toán phân loại chữ số viết tay mà nhóm sử dụng là mô hình CNN. Kết quả thực nhiệm được biểu diễn trong Hình 13.

Sau khi train xong mô hình phân loại chữ số, chúng tôi tiến hành detect ảnh sudoku về một ma trận 9x9 gồm các số có trong sudoku của ảnh.

Sau đó sử dụng một số hàm có sẵn của thư viện OpenCV, để làm giảm nhiễu ảnh: **cv.GausianBlu**r. Khi sử dụng hàm

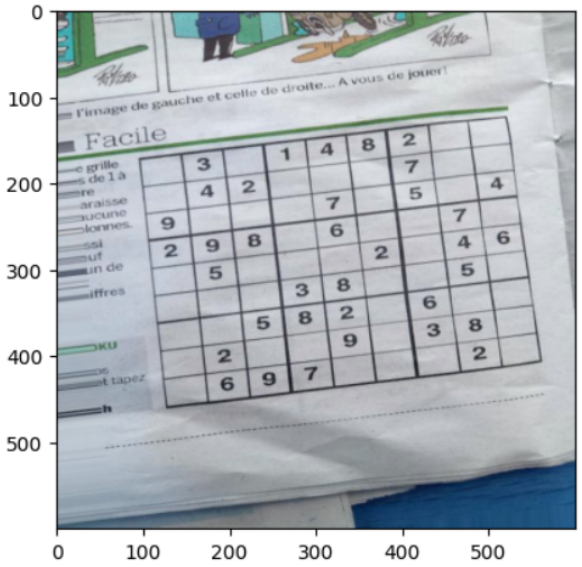


(a) Data MNIST (b) Data Chars74K

Hình 12: Data phân loại chữ số

| Train Accuracy | Validation Accuracy | Train Loss | Validation Loss | Test Accuracy | Test Loss |
|---|---|---|---|---|---|
| 0.9931 | 0.9923 | 0.024 | 0.029 | 0.989 | 0.036 |

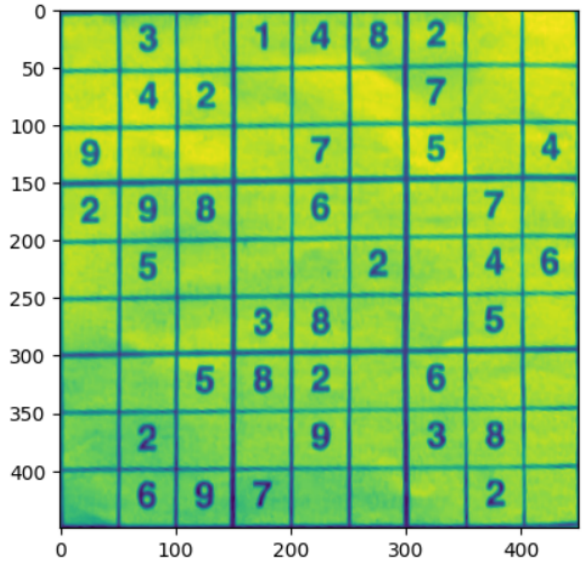
Hình 13: Độ chính xác của mô hình phân loại chữ số




Hình 14: Ví dụ về ảnh sudoku trong tập dữ liệu

này, bất kì cạnh sắc nét nào trong ảnh cũng sẽ được làm mịn và giảm thiểu hiện tượng nhòe viền. Tiếp theo, chúng tôi xác định ngưỡng cho mỗi pixel dựa trên vùng nhỏ xung quanh đó. Điều này mang lại kết quả tốt hơn cho hình ảnh có nhiều độ sáng khác nhau. Sau đó chúng tôi dùng hàm **findContours** để phát hiện hình dạng của sudoku. Nó trả về một mảng có các tọa độ (x, y) - là điểm biên của đối tượng. Sau khi tìm ra các điểm biên của đối tượng, chúng tôi tiếp tục đi tìm chu vi và diện tích của đường viền. Đường viền nào có diện tích lớn nhất và đường bao đóng thì đó chính là phạm vi của hình SudoKu. Tiếp đến chúng tôi sử dụng hàm **approxPolyDP** để bóp méo hình Sudoku về gần đúng thành hình vuông có kích thước 450x450.

Sau khi Detect được ảnh Sudoku, chúng tôi tiến hành append các các ô của sudoku vào một mảng, mỗi ô có kích thước là 50x50 (thành một mảng có kích thước 81x50x50).



Hình 15: Ảnh sau khi detect vùng chỉ chứa hình Sudoku

Sau đó, các ảnh con này sẽ được đưa vào mô hình phân loại chữ số để dự đoán số ở trong ô đó. Outpput của nó sẽ là một ma trận có kích thước 9x9 là các số đã được detect ra từ ảnh gốc ban đầu. Có được ma trận Sudoku xong, tiếp tục đưa nó vào mô hình giải sudoku để giải quyết bài toán.

Việc detect ảnh vẫn đang gặp nhiều khó khăn đối với những ảnh quá mờ hoặc ảnh bị mất đường viền bao quanh bảng sudoku. Không có BenMark để đánh giá.

## TÀI LIỆU


[1] Ileana-Diana V.D. Nicolae, Anca-Iuliana P.M. Nicolae, Limiting Backtracking in Fast Sudoku Solvers , 12 - 2018

[2] David Carmel, Solving Sudoku by Heuristic Search, Sep 17, 2021

[3] Brandon Amos, J. Zico Kolter, OptNet: Differentiable Optimization as a Layer in Neural Networks, 2-2018