A Rule Extraction Study Based on a Convolutional Neural Network

Guido Bologna[1,2(✉)]

[1] University of Applied Sciences and Arts of Western Switzerland, Rue de la Prairie 4, 1202 Geneva, Switzerland
[2] University of Geneva, Route de Drize 7, 1227 Carouge, Switzerland
Guido.Bologna@hesge.ch, Guido.Bologna@unige.ch

Abstract. Convolutional Neural Networks (CNNs) lack an explanation capability in the form of propositional rules. In this work we define a simple CNN architecture having a unique convolutional layer, then a Max-Pool layer followed by a full connected layer. Rule extraction is performed after the Max-Pool layer with the use of the Discretized Interpretable Multi Layer Perceptron (DIMLP). The antecedents of the extracted rules represent responses of convolutional filters, which are difficult to understand. However, we show in a sentiment analysis problem that from these "meaningless" values it is possible to obtain rules that represent relevant words in the antecedents. The experiments illustrate several examples of rules that represent n-grams.

Keywords: Convolutional Neural Networks · Rule extraction
Sentiment analysis

1 Introduction

A natural way to explain neural network responses is by means of propositional rules [8]. Andrews et al. introduced a taxonomy to characterize all rule extraction techniques [1]. Diederich and Dillon presented a rule extraction study on the classification of four emotions from SVMs [6]. Extracting rules in Sentiment Analysis (SA) is very challenging with thousands of words represented in the inputs [6]. As a consequence Diederich and Dillon restricted their study to only 200 inputs and 914 samples. Bologna and Hayashi used the Quantized Support Vector Machine to generate rules explaining Tweets' sentiment polarities [4]. In this work we propose a model to generate rules from a Convolutional Neural Network architecture (CNN) that is trained with a dataset related to SA. To the best of our knowledge this problem has not been tackled.

Since Convolutional Neural Networks (CNNs) started to be broadly used less than ten years ago, very few works have proposed to determine the acquired knowledge in these models. Several authors proposed to interpret CNNs in the neighborhood of the instances. As an example, Ribeiro et al. presented LIME

A. Holzinger et al. (Eds.): CD-MAKE 2018, LNCS 11015, pp. 304–313, 2018.
https://doi.org/10.1007/978-3-319-99740-7_22

whose purpose is to learn an interpretable model in the local region close to an input instance [13]. Koh et al. determined the training instances that are the most important for the prediction [10]. Finally, Zhou et al. presented CAM whose purpose is to determine discriminative image regions using the average pooling of CNNs [14].

This work illustrates rule extraction from a CNN architecture shaped by an input layer representing sentences of word embeddings [11], a convolutional layer with filters, and max-pooling layer followed by a fully connected layer. After the training phase, the fully connected layer is replaced by a *Discretized Interpretable Multi Layer Perceptron* (DIMLP) [2] that allows us to extract rules. This subnetwork approximates the fully connected part of the original CNN to any desired precision. Nevertheless, the antecedents of the extracted rules represent maximal responses of convolutional filters, which can not be understood directly. Thankfully, it is possible to go back to the words that are relevant in the decision-making process. Specifically, these words are structured into n-grams, that depend on the size of the convolutional filters. In the following sections we first describe the used model, then we present the experiments, followed by the conclusion.

2 The Interpretable CNN Architecture

The CNN architecture used here is similar to those typically used in SA problems [9]. Before rule extraction, we train a CNN having only a fully connected layer between the last two layers of neurons. Then, we perform rule extraction by replacing these last two layers of neurons by a special subnetwork that can be translated into symbolic rules [2]. Hence, after training the modified CNN for rule extraction has two components:

- a CNN without the fully connected layer;
- a DIMLP with only a hidden layer that approximates the CNN fully connected layer.

Rules generated from the DIMLP component are related to filter thresholds in the convolutional layer of the CNN. From these values relevant words represented in the input layer can be determined. More details are given in the following paragraphs.

2.1 The CNN Subnetwork

Our CNN subnetwork is quite similar to that proposed in [5,9]. A smaller representation of the adopted CNN architecture is described in Fig. 1. The inputs are words represented by word embeddings of dimensionality $d = 300$, calculated with *word2vec* [11]. Here, the maximal number of words in a sentence is equal to $s = 59$, thus the size of an instance in the left part of Fig. 1 is equal to $17700 (= 59 * 300)$. Then, an instance is convolved with filters of different size. Specifically, for each filter we define a filtering matrix of size $f \cdot d$, with $f = 1, 2, 3,$

corresponding to the number of words on which it operates. After convolution we apply a *Relu* function ($\text{Relu}(x) = \text{Max}(0, x)$), which is non-linear. For each filter, the result of the convolution provides a vector of size $s - f + 1$. Afterward, the max-pooling operator is applied and the maximal value is retained, independently of where it is located in a sentence. All the max-pooling units are concatenated (right most layer in Fig. 1) and then follows a fully connected layer. In the output layer, a Softmax activation function is used. Specifically, for N a_i scalars it calculates an N-dimensional vector with values between 0 and 1. It is given as:

$$S_i = \frac{\exp(a_i)}{\sum_{k=1}^{N} \exp(a_k)}; \qquad \forall i \in 1 \ldots N. \tag{1}$$

We use the Lasagne Library Package [7] to train our CNNs. The training phase is achieved with the Cross-Entropy loss function. After learning the fully connected layer, which is not represented in Fig. 1 is replaced by a DIMLP subnetwork having two layers of weights, with the second being equal to the fully connected layer of the CNN.

2.2 The DIMLP Subnetwork

DIMLP differs from standard Multi Layer Perceptrons in the connectivity between the input layer and the first hidden layer. Specifically, any hidden neuron receives only a connection from an input neuron and the bias neuron, as shown in Fig. 2. After the first hidden layer, neurons are fully connected. Note that very often DIMLPs are defined with two hidden layers; the number of neurons in the first hidden layer being equal to the number of input neurons. The key idea behind rule extraction from DIMLPs is the precise localization of axis-parallel discriminative hyperplanes. In other words, the input space is split into hyper-rectangles representing propositional rules. Specifically, the first hidden layer creates for each input variable a number of axis-parallel hyperplanes that are effective or not, depending on the weight values of the neurons above the first hidden layer. More details on the rule extraction algorithm can be found in [3].

The activation function in the output layer of a standard DIMLP [2] is a sigmoid function given as

$$\sigma(x) = \frac{1}{1 + \exp(-x)}. \tag{2}$$

However, here since the CNN is trained with a Softmax function in the output layer, we replace the sigmoid by it. In the first hidden layer the activation function is a staircase function $S(x)$ with Q stairs that approximate the Identity function $I(x) = x$; $\forall x \in (R_{min}..R_{max})$, with two constants R_{min} and R_{max}.

$$S(x) = R_{min}, \text{ if } x \leq R_{min}; \tag{3}$$

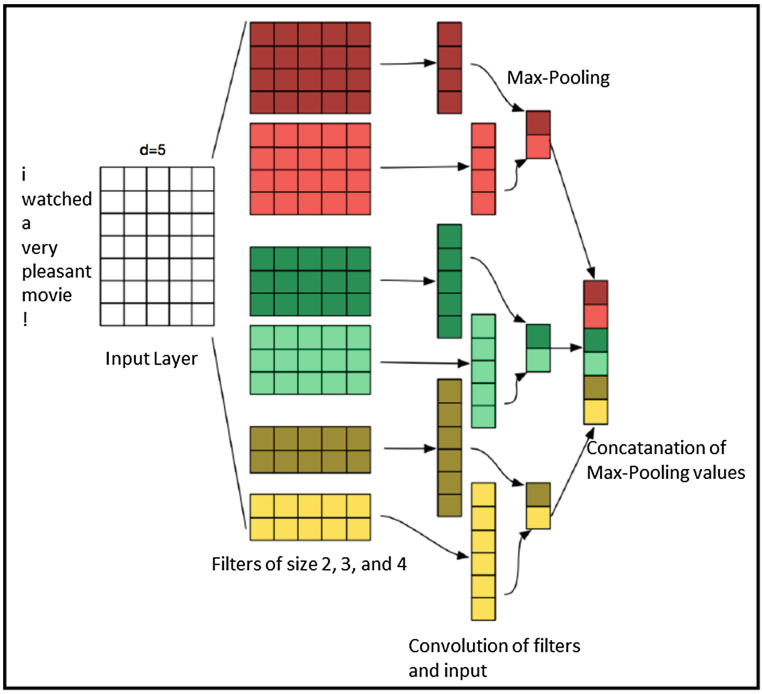

Fig. 1. The CNN subnetwork has an input layer with data provided as word embeddings; these word vectors are represented horizontally, while sentences are represented vertically. Then follows a convolutional layer with filters of different size. The convolution of the input with filters provides one-dimensional vectors (represented vertically). Finally, the "max" function reduces the size of these vectors that are finally concatanated (last layer on the right). The fully connected layer is not represented.

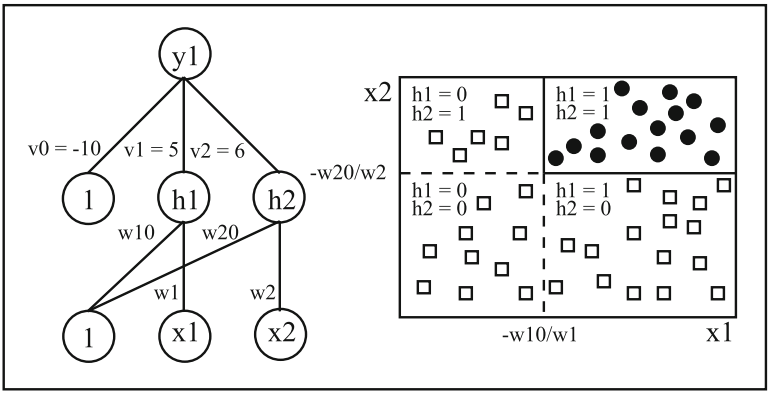

Fig. 2. A DIMLP network creating two discriminative hyperplanes. The activation function of neurons h_1 and h_2 is a step function, while for output neuron y_1 it is a sigmoid.

R_{min} represents the abscissa of the first stair. By default $R_{min} = 0$.

$$S(x) = R_{max}, \text{ if } x \geq R_{max}; \tag{4}$$

R_{max} represents the abscissa of the last stair. By default $R_{max} = 1$. Between R_{min} and R_{max} $S(x)$ is given as

$$S(x) = I(R_{min} + \left[q \cdot \frac{x - R_{min}}{R_{max} - R_{min}}\right](\frac{R_{max} - R_{min}}{q})). \tag{5}$$

Square brackets indicate the integer part function and $q = 1, \ldots Q$. The step function $t(x)$ is a particular case of the staircase function with only one step:

$$t(x) = \begin{cases} 1 \text{ if } x > 0; \\ 0 \text{ otherwise.} \end{cases} \tag{6}$$

The approximation of the Identity function by a staircase function depends on the number of stairs. The larger the number of stairs the better the approximation. Note that the step/staircase activation function makes it possible to precisely locate possible discriminative hyperplanes.

As an example, in Fig. 2 assuming two different classes, the first is being selected when $y_1 > \sigma(0) = 0.5$ (black circle) and the second with $y_1 \leq \sigma(0) = 0.5$ (white squares). Hence, two possible hyperplane splits are located in $-w_{10}/w_1$ and $-w_{20}/w_2$, respectively. As a result, the extracted unordered rules are:

- $(x_1 < -w_{10}/w_1) \rightarrow$ square
- $(x_2 < -w_{20}/w_2) \rightarrow$ square
- $(x_1 \geq -w_{10}/w_1)$ and $(x_2 \geq -w_{20}/w_2) \rightarrow$ circle

2.3 From Meaningless Rules to Meaningful Rules

Each antecedent extracted from the max-pooling-layer is given as $a < t$, or $a \geq t$; t being a filter threshold involving in the input layer a number of activated words. Specifically, these words correspond to bigrams and trigrams when the filter size in the convolutional layer is equal to two or three, respectively. In practice, filters of size equal to one are convolved with all possible single words. Then with the obtained values, we retain all the single words that make true a rule antecedent related to the max-pool-layer. This is repeated for filters of size two with respect to bigrams and so on with other filter sizes.

Generally, the condition expressed in a rule antecedent is true with more than one n-gram; thus a rule antecedent which is true for the DIMLP subnetwork implies a disjunction of one or more n-grams represented in the input layer (one or more n-grams connected by a logical or). Nevertheless, with the use of the "Max" function a unique n-gram becomes dominant (the one with the highest activation) and cancels the others.

Disjunctions of n-grams related to a rule antecedent extracted from the max-pooling-layer involving thousands of words could be considered too numerous.

However, in practice these words are not necessarily encountered, especially for rules activated by a small number of examples. These words will be useful for determining possible contradictions, such as the simultaneous presence of words/n-grams that are clearly of positive polarity and others of negative polarity.

3 Experiments

In the experiments we use a well-known binary classification problem describing movie reviews with Tweets [12].[1] The number of examples is equal to 10662, with half of them belonging to the positive class. Note that the total number of single words in the dataset is equal to 21426, while the total number of bigrams is equal to 111590. Words are coded into vectors of word embeddings of size 300 [11]. Two CNN architectures were defined; one with 50 filters of size one and two ($f = 1, 2$) and another with 40 filters of size one, two and three. Hence, the last layer in Fig. 1 has 100 and 120 neurons, respectively. For deep learning training we use Lasagne libraries, version 0.2 [7]. The loss function is the categorical cross-entropy and the training parameters are:

- learning rate: 0.02;
- momentum: 0.9;
- dropout = 0.2;

From the whole dataset we selected the first 10% of the samples of each class as a testing set and the rest as training examples. Moreover, a subset of the training set representing 10% of it was used as a tuning set for early-stopping. Table 1 shows the results for the first CNN architecture. The first row of this Table is related to the original CNN, while the other rows provide results of the approximations obtained with the CNN-DIMLP combination by varying the number of stairs in the staircase activation function. Columns from left to right designate:

- train accuracy;
- predictive accuracy on the testing set;
- fidelity, which is the degree of matching between rules and the model;
- predictive accuracy of the rules;
- predictive accuracy of the rules when rules and model agree;
- number of extracted rules and total number of rule antecedents.

Note that these rules involve filter responses in the antecedents from which n-grams are determined (cf. Sect. 2.3). Table 2 presents the results obtained with the second CNN architecture.

The best predictive accuracy of rules was obtained with the second architecture, which takes into account trigrams. Note however that accuracy performance in this work is not a priority, since our purpose is to demonstrate how to generate

[1] Available at: http://www.cs.cornell.edu/people/pabo/movie-review-data/.

Table 1. Results obtained with the fist CNN architecture including filters of size one and two, involving unigrams and bigrams. In the DIMLP subnetwork the number of stairs of the staircase activation function is varied from 20 to 200.

	Train Acc.	Test Acc.	Fidelity	Test Acc. (r1)	Test Acc. (r2)	#Rules/#Ant
CNN	**81.5**	**74.6**	–	–	–	–
CNN (q=20)	81.2	74.0	93.6	72.9	75.0	747/6159
CNN (q=50)	**81.5**	74.0	94.8	**74.3**	**75.5**	636/5730
CNN (q=100)	81.4	74.1	95.3	73.0	74.7	669/5619
CNN (q=200)	81.4	74.5	**95.8**	73.4	75.0	562/**5131**

Table 2. Results obtained with the second CNN architecture including filters of size one, two and three, involving unigrams, bigrams and trigrams.

	Train Acc.	Test Acc.	Fidelity	Test Acc. (r1)	Test Acc. (r2)	#Rules/#Ant
CNN	**83.0**	**75.4**	–	–	–	–
CNN (q=20)	**83.0**	75.0	94.6	73.4	75.6	637/5633
CNN (q=50)	82.9	75.3	95.2	72.8	75.3	559/5065
CNN (q=100)	82.9	**75.4**	**95.5**	**75.2**	**76.5**	568/4885
CNN (q=200)	**83.0**	75.3	94.0	73.5	75.9	555/**4798**

meaningful propositional rules from CNNs. Overall, fidelity on the testing set is above 90%, meaning that rules explain CNN responses in a large majority cases.

The antecedents of rules extracted from the DIMLP subnetwork involve long lists of n-grams. Here we illustrate several rules extracted from the first architecture with $q = 200$ (the one with the highest fidelity). Rules are ranked according to their support with respect to the training set, which is the proportion of covered samples. Note that rules are not disjointed, which means that a sample can activate more than a rule. The first rule has a support of 765 training samples and 76 testing samples; it is given as:

– $(f_{51} < 0.285)$ $(f_{65} < 0.09)$ $(f_{70} < 0.275)$ $(f_{74} \geq 0.185)$ $(f_{87} < 0.305)$ $(f_{99} < 0.06)$ Class = POSITIVE

Here, f designates maximal values of filters with respect to the max-pooling layer. Indexes between one and 50 are related to single words, while those between 51 and 100 correspond to bigrams. The accuracy of this rule is 93.9% on the training set and 88.1% on the testing set. Antecedent $f_{74} \geq 0.185$ involves the presence of at least one bigram in a list of 1218 possible bigrams. Generally, the possible bigrams are not necessarily encountered in the tweets activating a rule. By negating all other antecedents of this rule we obtain a list of 12730 bigrams that are required to be absent.

We illustrate a number of sentences with possible bigrams including the dominant bigram, represented in bold and related to the rule shown above. Note that three consecutive words in bold represent two consecutive bigrams:

1. offers **that rare** combination of entertainment **and education**.
2. **a thoughtful**, provocative, insistently humanizing film.

3. **a masterful** film from a master filmmaker**, unique** in its deceptive grimness**, compelling** in its fatalist worldview.
4. cantet **perfectly captures** the hotel lobbies, two-lane highways, and roadside cafes that permeate vincent's days.
5. the film makes **a strong** case for the importance of the musicians in creating the motown sound.
6. **a compelling coming-of-age** drama about the arduous journey of **a sensitive** young girl through a series of foster homes and **a fierce** struggle to pull free from her dangerous and domineering mother's hold over her.
7. this delicately observed story**, deeply** felt **and masterfully** stylized, is **a triumph** for its maverick director.
8. **a portrait** of alienation **so perfect**, it will certainly succeed in alienating most viewers.
9. it's sincere to a fault, but, unfortunately, not **very compelling** or much fun.

The first eight tweets are correctly classified and we can clearly recognize words of positive polarity. The two tweets at the end of the list are classified as positive, whereas their class is negative. The last tweet is wrongly classified, because "not" before "very compelling" has been ignored. Regarding the ninth tweet, "so perfect" contributes without any doubt to a positive polarity; then, at the end of the sentence "alienating" contributes to its true negative classification, but it is not considered at all by the rule.

Rule number 17 is given as:

– $(f_7 < 0.315)$ $(f_{39} < 0.295)$ $(f_{54} \geq 0.05)$ $(f_{65} < 0.12)$ $(f_{75} < 0.25)$ $(f_{83} < 0.13)$ $(f_{95} \geq 0.135)$ $(f_{96} \geq 0.16)$ Class = POSITIVE

It has a support of 404 samples in the training set and 47 samples in the testing set with an accuracy of 91.6% and 83.0%, respectively. The presence of one or more bigrams is imposed by f_{54}, f_{95}, and f_{96}; it contains 8266 elements. Ninety-six single words must be absent; they are related to f_7, and f_{39}. Moreover, 6307 mandatory absent bigrams depends on f_{65}, f_{75}, and f_{83}. Ten tweets with their possible bigrams including the dominant bigram are shown here; the last two tweets are wrongly classified:

1. **an utterly compelling** 'who wrote it' in which the reputation of **the most** famous author who ever lived comes into question.
2. between the drama of cube? s personal revelations regarding what the shop means in the big picture, iconic characters gambol fluidly through the story, with **charming results**.
3. it's **an old** story, but **a lively** script, sharp acting and **partially animated** interludes make just **a kiss** seem **minty fresh**.
4. this is simply **the most fun** you'll ever have with **a documentary**!
5. one of **the best**, most **understated performances** of [jack nicholson's] career.
6. tadpole is **a sophisticated, funny and good-natured** treat, slight but **a pleasure**.
7. **a smart, sweet and playful romantic** comedy.
8. mr . parker has **brilliantly updated** his source and **grasped its** essence, composing **a sorrowful and hilarious tone** poem about alienated labor, or **an absurdist** workplace sitcom.

9. **beautifully filmed and well acted.** . . but admittedly problematic in its narrative specifics.
10. one of the oddest **and most** inexplicable sequels in movie history.

The following list illustrate negative tweets correctly classified by a rule reaching an accuracy of 100% on the testing set:

1. it's an 88-min highlight reel that's **86 min too long**.
2. such an incomprehensible mess that it **feels less like bad** cinema **than like being** stuck in a dark pit having a nightmare **about bad** cinema.
3. during the **tuxedo's 90 min of** screen time, **there isn't** one true 'chan moment'.
4. **the script becomes** lifeless and falls apart like a cheap lawn chair.
5. **the script falls** back on **too many** tried-and-true shenanigans that hardly distinguish it from **the next teen** comedy.
6. a close-to-solid espionage thriller with the misfortune of being released a **few decades too late**.

Finally, we show another example of negative tweets correctly classified by another rule yielding predictive accuracy equal to 100%:

1. maybe leblanc thought, "hey, the movie about the baseball-playing monkey **was worse**."
2. a muddled limp biscuit of a movie, a **vampire soap opera that doesn't make** much **sense even** on its **own terms**.
3. the **script becomes lifeless** and falls apart like a cheap lawn chair.
4. a baffling subplot involving smuggling drugs inside danish cows **falls flat**, and if you're going to alter the bard's ending, you'd **better have** a good alternative.
5. given the **fact that virtually no one** is bound to show up at theatres for it, the project **should have been** made for the tube.
6. jonathan parker's bartleby **should have been** the be-all-end-all of the modern-office anomie films.

4 Conclusion

We presented a model that allowed us to extract rules of high fidelity from a typical trained CNN architecture for Sentiment Analysis. Rule extraction was first applied to the layer before the output layer and then relevant words were determined from convolutional filters thresholds. Generated rules are described by disjunctions of n-grams that must be present or conjunctions of n-grams that must be absent. Moreover, extracted n-grams do not depend on particular positions in sentences. In the experiments, several examples of tweets with discriminatory bigrams that explained CNN responses were illustrated. These discriminatory words are important, as they can be used to understand how correct/wrong classifications are obtained by the classifier.

References

1. Andrews, R., Diederich, J., Tickle, A.B.: Survey and critique of techniques for extracting rules from trained artificial neural networks. Knowl.-based Syst. **8**(6), 373–389 (1995)
2. Bologna, G.: Rule extraction from a multilayer perceptron with staircase activation functions. In: Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks, 2000, IJCNN 2000, vol. 3, pp. 419–424. IEEE (2000)
3. Bologna, G.: A model for single and multiple knowledge based networks. Artif. Intell. Med. **28**(2), 141–163 (2003)
4. Bologna, G., Hayashi, Y.: A rule extraction study from svm on sentiment analysis. Big Data Cogn. Comput. **2**(1), 6 (2018)
5. Cliche, M.: Bb_twtr at semeval-2017 task 4: Twitter sentiment analysis with cnns and lstms. arXiv preprint arXiv:1704.06125 (2017)
6. Diederich, J., Dillon, D.: Sentiment recognition by rule extraction from support vector machines. In: CGAT 09 Proceedings of Computer Games, Multimedia and Allied Technology 09. Global Science and Technology Forum (2009)
7. Dieleman, S., et al.: Lasagne: First release, August 2015. https://doi.org/10.5281/zenodo.27878
8. Holzinger, A., Biemann, C., Pattichis, C.S., Kell, D.B.: What do we need to build explainable AI systems for the medical domain? arXiv preprint arXiv:1712.09923 (2017)
9. Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014)
10. Koh, P.W., Liang, P.: Understanding black-box predictions via influence functions. arXiv preprint arXiv:1703.04730 (2017)
11. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
12. Pang, B., Lee, L.: A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. In: Proceedings of the 42nd annual meeting on Association for Computational Linguistics, p. 271. Association for Computational Linguistics (2004)
13. Ribeiro, M.T., Singh, S., Guestrin, C.: Why should i trust you?: Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1135–1144. ACM (2016)
14. Zhou, B., Bau, D., Oliva, A., Torralba, A.: Interpreting deep visual representations via network dissection. arXiv preprint arXiv:1711.05611 (2017)