Dynamic CNN Models For Fashion Recommendation in Instagram

Shatha Jaradat, Nima Dokoohaki, Kim Hammar, Ummal Wara, Mihhail Matskin
Department of Software and Computer Systems
KTH - Royal Institute of Technology
Stockholm, Sweden
{shatha,nimad,kimham,ummul,misha}@kth.se

***Abstract*—Instagram as an online photo-sharing and social-networking service is becoming more powerful in enabling fashion brands to ramp up their business growth. Nowadays, a single post by a fashion influencer attracts a wealth of attention and a magnitude of followers who are curious to know more about the brands and style of each clothing item sitting inside the image. To this end, the development of efficient Deep CNN models that can accurately detect styles and brands have become a research challenge. In addition, current techniques need to cope with inherent fashion-related data issues. Namely, clothing details inside a single image only cover a small proportion of the large and hierarchical space of possible brands and clothing item attributes. In order to cope with these challenges, one can argue that neural classifiers should become adapted to large-scale and hierarchical fashion datasets. As a remedy, we propose two novel techniques to incorporate the valuable social media textual content to support the visual classification in a dynamic way. The first method is adaptive neural pruning (*DynamicPruning*) in which the clothing item category detected from posts' text analysis can be used to activate the possible range of connections of clothing attributes' classifier. The second method (*DynamicLayers*) is a dynamic framework in which multiple-attributes classification layers exist and a suitable attributes' classifier layer is activated dynamically based upon the mined text from the image. Extensive experiments on a dataset gathered from Instagram and a baseline fashion dataset (DeepFashion) have demonstrated that our approaches can improve the accuracy by about 20% when compared to base architectures. It is worth highlighting that with Dynamiclayers we have gained 35% accuracy for the task of multi-class multi-labeled classification compared to the other model.**

***Index Terms*—Neural Pruning; Dynamic Computation Graph; Dynamic CNN; Image Classification; Text Mining; Fashion Recommendation**

I. Introduction

Fashion in the age of Instagram is changing the way in which clothing items are presented and even the way they are designed [1]. Millions of users post photos of their "outfit of the day - #ootd", receive questions from other users about the outfits, and market the fashion brands they wear. A fashion look in Instagram can attract many customers to the brands that are available in the post [2]. Sponsors invest in the details of outfits of their fashion influencers not just to impress the followers, but as an investment to attract more customers to purchase the outfits. With the appropriate hashtags, fashion retailers increase their findability and incorporate more followers into their business. Instagram is a great platform to analyze lifestyle trends. Its powerful analytics system uses advanced AI and Big Data technologies to process the massive amounts of data created by users, along with their search preferences, in order to feed their ads systems [3]. This in turn yields great returns to brands who are actively looking to be found by users with certain interests. As the quantity and types of data increase, this personalization task becomes more complicated. Hence, a more accurate detection of users' favorite styles and brands can open up novel possibilities for enhancing online shopping recommendations and advertisements in Instagram.

Although characterized as an image-sharing platform, Instagram also hosts large volumes of unstructured user-generated text. Specifically, an Instagram post can be associated with an image caption written by the author of the post, by comments written by other users, and by hashtags in the image that refer to other users or brands. In the context of fashion recommendation, we performed a case study on Instagram posts in the fashion community and found that clues about the image contents can be found in the associated text. For instance, the author of the post might describe the contents of the image, or users might comment about details in the image. An example from our text analysis output is shown in **Figure 1**. More details about the text mining methodology that we followed are illustrated in **Section 3.1**.

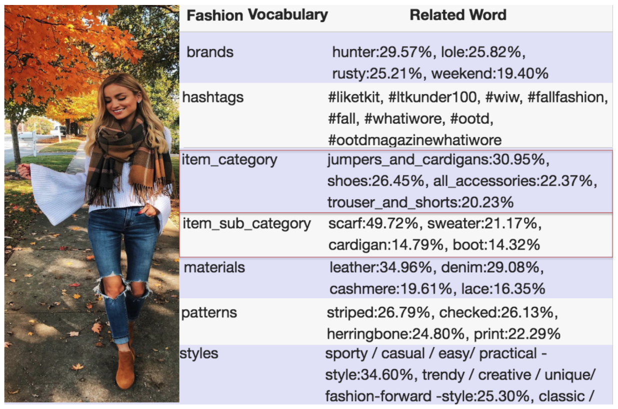
As part of the implementation of our fashion and style recommendation framework described in [4] we have the task of classifying an image into a large space of possible fashion brands, clothing categories, sub-categories and style attributes. For example: Dress (category) - Casual (sub-category) - Satin, Floral (Attributes) from Zara (Brand)[1]. Given that there might be more than one clothing item in the image, the task requires a multi-class multi-labeled classification. To tackle these challenges, dynamic models need to be introduced to handle different types of clothing categories. Recently, deep learning frameworks such as Pytorch[2] and Tensorflow Fold[3] have started to support Dynamic Computational Graphs (DCGs), which are also known as "define by run" graphs. With DCG support, a deep learning architecture graph can be defined at runtime, and then backpropagation can use the dynamically

[1]We have referred to Zalando.com; an online shopping website to define the structure in our ontology.
[2]https://pytorch.org/
[3]https://github.com/tensorflow/fold

built graph [5]. This simplifies the implementation of deep-learning models that operate over data of varying size or data that have a different structure based on input. Introducing dynamic computational graphs can open up the way to many novel inference algorithms to speed up the training by taking advantage of these flexible models. In our solution, we have multiple ”sub-categories” and ”attributes” layers, each related to a certain clothing category, and different categories might have different attributes. Having strong hints from text analysis about the possible categories, sub-categories and attributes with percentages can help us to choose the possible sub-category and attributes' classification layers to connect to dynamically according to the text. In our work, we present ***DynamicCNN***: a novel application of DCGs using text mining.

Deciding the size of the neural network can greatly affect the network generalization and inference speed. Large systems might learn very slowly and might be sensitive to the initial parameters. Neural pruning is one possible way to control the number of redundant network parameters or feature maps that have less contribution to the prediction's output [6]. This consequently affects the network size and possibly the inference speed. Finding a dynamic approach to decide the parameters that are more or less important for solving certain tasks can greatly help in this context. Usually, pruning identifies the unnecessary parameters by calculating the sensitivity of the error to their removal, but in our case we have used the detected text from our text analysis in order to control dynamically the parameter connections that can be deactivated for certain inputs (our ***DynamicPruning*** model). Continuing on the clothing classification example: instead of having more than one sub-category or attribute layers, we experimented having one large sub-category layer with all values from all possible sub-categories, and another layer that combines all possible attributes from different clothing categories. Then, we developed a dynamic approach to dynamically deactivate the parameter connections that are irrelevant to the detected category in the image. This process is done dynamically based on the detected category in the image.

Our contributions in this paper are two novel techniques (*DynamicPruning* and *DynamicLayers*) to incorporate the social media textual content to support the visual classification in neural networks in a dynamic way. We evaluated our models on two large fashion-specialised datasets. Our experiments demonstrated the improvements achieved by both models in enhancing classification accuracy. In this paper, we evaluate the proposed techniques on basic CNN architectures to highlight the added value of text integration in such model and to pave the way for integrating text support with our proposed architecture described in [4]. Our evaluation of the mentioned architecture will be in comparison to the popular related models such as [7].



Fashion Vocabulary	Related Word
brands	hunter:29.57%, lole:25.82%, rusty:25.21%, weekend:19.40%
hashtags	#liketkit, #ltkunder100, #wiw, #fallfashion, #fall, #whatiwore, #ootd, #ootdmagazinewhatiwore
item_category	jumpers_and_cardigans:30.95%, shoes:26.45%, all_accessories:22.37%, trouser_and_shorts:20.23%
item_sub_category	scarf:49.72%, sweater:21.17%, cardigan:14.79%, boot:14.32%
materials	leather:34.96%, denim:29.08%, cashmere:19.61%, lace:16.35%
patterns	striped:26.79%, checked:26.13%, herringbone:24.80%, print:22.29%
styles	sporty / casual / easy/ practical - style:34.60%, trendy / creative / unique/ fashion-forward -style:25.30%, classic /

Fig. 1. Text analysis results from our system. In the picture, the system detects that the fashionista wears a jumper (Jumpers and cardigans), a scarf (all accessories), a boot (shoes), and trousers (trouser and shorts) as in the category row.

II. Related Work

In this section, we will briefly present an overview of the previous work related to large-scale and multi-label classification and pruning neural networks.

A. Deep Learning For Large-Scale Multi-Labeled Image Classification

With the continuous increase in the amounts of visual and textual data, the need for scalable methods of classification in machine learning is becoming increasingly important. Many approaches which have been proposed for this purpose depend on label space compression, such as: [8] and [9]. The main idea in such approaches is performing predictions on a smaller dimensional label space and then recovering the labels to their original higher dimension. A recent example of a label compression solution in a generic empirical risk minimization (ERM) framework is [9]. Other approaches depend on feature space reduction such as: [10], where they apply Canonical Correlation Analysis for dimensionality reduction. Focusing on deep neural networks multi-label classification, [11] and [12] are examples of work that improved pairwise ranking for the purpose of multi-label image classifications, where the objective is to maximize ranks of positive labels. An example of scalable neural network design is proposed in [13] for the purpose of achieving the quick classification of high-dimensional and large data. The classification speed in their model is achieved by adding an extra layer for clustering learned features into large clusters and activating only one or a few clusters.

B. Pruning Convolutional Neural Networks For Efficient Resource Usage

Generally, pruning algorithms can be classified into two groups: a group that calculates the sensitivity of network errors with respect to the removal of certain parameters or feature maps, and another group that modifies the error function in a way that rewards the network for choosing an efficient combination of parameters [14]. Practically, pruning is done in an iterative way, where the weights/feature maps are ranked according to how much they contribute, and the low ranking ones are removed. Then, the network is fine-tuned and based on the network loss, this process can be repeated again [6]. One challenge in pruning methods is that they do not necessarily detect correlated elements. Thus, the criterion of choosing neuron-ranking methods and the speed of pruning are essential, as they might result in a big drop in accuracy, and a longer training time of the network to recover. The ranking of neurons can be done according to the L1/L2 regularization mean of neuron weights, their mean activations, the number of times a neuron was not zero, and some other methods. Different strategies for pruning weights were proposed in [15], and for pruning entire filters (feature maps) in [16]. Usually, ranking methods are to prune filters, and then observe how the cost function changes when running on the training set. But, in our approach, we dynamically choose the parameter connections based on the identified category from text which is our criteria to decide the importance of connections.

III. Methodology

In this section, we describe the methodology we have followed for text mining, and for our two proposed methods: adaptive neural pruning and dynamic CNN.

A. Text Mining

We have formulated the task of extracting fashion details from the text associated with Instagram posts as a mixture between an information extraction and a ranking problem. As text on Instagram often is multi-lingual and *noisy*, information extraction using linguistic rules is fragile. For this reason, we have made use of word embeddings [17] as a central component to extract fashion details from Instagram posts. The extraction is carried out as follows. First, the text is tokenized with NLTK's [4] TweetTokenizer, that is designed to recognize text from social media. Then the text is normalized by removing stopwords, lemmatizing and lower-casing all tokens, extracting hashtags, emojis, and user-handles using regular expressions, and segmenting hashtags. Next, we use word embeddings trained on a large corpora of Instagram text to semantically relate the tokens in the text to our manually predefined fashion ontology that contains brands, items, patterns, materials, and styles. Mathematically, we match tokens to the ontology based on the cosine similarity between the tokens in the text and terms in the ontology. In this mapping, we combine the cosine similarity with several other factors using a linear combination that includes the Term Frequency-Inverse Document Frequency (TF-IDF), a term-score, and a lookup in the Probase API [5] for ambiguity resolution. Finally, after this mapping between the input text and the ontology, the terms in the ontology are ranked based on their accumulated matching with the input text.

[4]https://www.nltk.org/

[5]http://haixun.olidu.com/probase.html

B. Proposed Methods

Basic Formulation: We consider two possible cases in image classification: single-class multi-labeled and multi-class multi-labeled. A formal description of both cases is given as follows: Let D = $\{xi, Ci, Si, Ai\}_{i=1}^{N}$ be a dataset with xi is the ith image, N is the total number of images, $Ci \subseteq C$ is the category set associated with the image, where C = $\{1, 2, .., K\}$ is the set of all category indices, and K is the total number of categories. Examples of categories are: coats and dresses. $Si \subseteq S$ is the subcategory set, where S = $\{1, 2, .., M\}$ is the set of subcategory indices and M is the total number of subcategories. Examples of subcategories of the dresses category are: casual and business. $Ai \subseteq A$ is the corresponding attribute set, where A = $\{1, 2, .., P\}$ is the set of attribute indices and P is the total number of attributes. Examples of attributes are: floral and striped. Each category C has a different set of related attributes. For example, some attribute groups for dresses category are: length, collar-type, material, pattern. On the other hand, heel height, fastener, heel type, toe are examples of attribute groups that belong to shoes category. We have two types of annotated images. A group of *single-class multi-labeled* images in which each image xi is tagged with one category $|Ci| = 1$ and a set of multiple attributes Ai.The other group is the *multi-class multi-labeled* images in which each image xi is tagged with a set of subcategories Si and a set of categories Ci and a set of multiple attributes Ai. **Figure** 2 (a) illustrates the basic architecture used to describe our adaptive neural pruning model. For single-class multi-labeled scenarios, we define two linear layers: (a) category linear layer that produces a tensor $\hat{C}$ of size $k = K$ representing the scores of predicted categories, (b) attributes linear layer that produces a tensor $\hat{A}$ of size $p = P$ presenting the confidence scores of attributes predictions. We use RELU as activation function and Mean Squared Error (MSE) for loss computation. MSE is defined as: MSE = $\frac{1}{n}(Yi - \hat{Y}i)^2$ where Ŷ is the vector of predictions and Y is the vector of observed values of the variable being predicted (ground truth). For multi-class mulitlabeled scenarios, we add subcategory linear layer that produces $\hat{S}$ of size $m = M$ that presents the confidence scores of subcategories. Below is a detailed description of our proposed methods.

Adaptive Neural Pruning: for classification tasks that involve a large number of possible choices, the computations in fully connected layers become more complicated and time-consuming. However, with the ability to get useful hints from text mining, it becomes possible to control the choices dynamically. Hence, the amount of connections between layers can be minimized. With this approach, we set the weights of irrelevant connections to zero. In other words, we customize the connections according to their importance in connection with the detected item (from text mining). As we previously mentioned, we consider two possible cases in image classifications and we will use clothing category and attributes as an

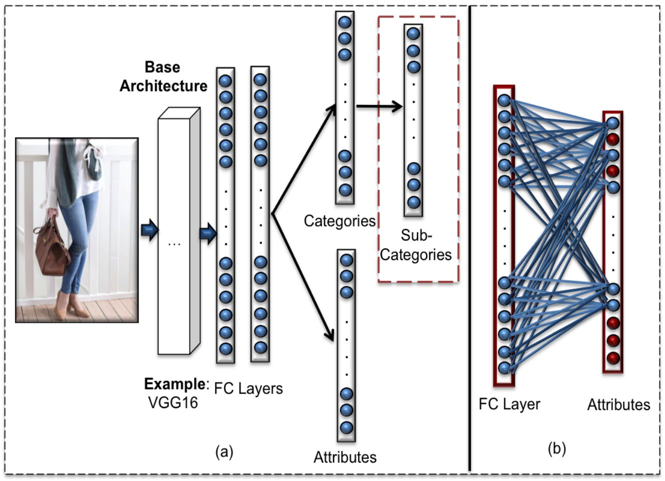


Fig. 2. (a) A basic architecture, such as VGG16 (except the last FC layers, followed by 2 new FC layers, a layer for categories, and a layer for attributes predictions). For multi-class scenario, we have the sub-categories layer in addition. (b) DynamicPruning architecture - dynamic connections are decided based on some of the text tag(s) associated with the image. The irrelevant attributes to the detected **categories** are deactivated (zero weight)

example to explain the general approach.

TABLE I
LIST OF SYMBOLS USED IN THE ALGORITHM 1 AND 2

Symbol	Meaning
Ng	Number of nodes in the first level of classification. In our example: number of categories.
Ns	Number of nodes in the more specific level of classification. In our example: number of attributes or sub-categories.
No	Number of neurons from previous output layer
α	Initial weight increase value for relevant attributes (direct hints from text)
β	Initial weight increase value for the relevant **range** of attributes (by using category to decide the possible range)
Tz	Tensor of size equivalent to number of neurons in previous layer for weight connections filled with zeros
$T\beta$	Tensor of size equivalent to number of neurons in previous layer for weight connections increased with value β
$T\alpha$	Tensor of size equivalent to number of neurons in previous layer for weight connections increased with value α
La	List containing specific values per each general level of classification. For example, list of subcategories per category.
Ai	Number of attributes/subcategories per specific category. Equivalent to Lai

1) Single-class Multi-labeled Scenarios: We use the clothing's item category detected from text mining as a supporting factor to decide the irrelevant attributes connections. So, the irrelevant connections weights are set to zero, while the relevant range of connections weights is adjusted with a certain value β. As mentioned before in section B: for each category there is a specific set of relevant attribute groups, and each attribute group contains many values. The algorithm is illustrated in **Algorithm** 1, where we show in details the mechanism on how we customize the connections upon the matching with a certain category. **Table I** presents the common symbols used in **Algorithms** 1 and 2. Line 9 in Algorithm 1 describes setting the weight of connections of irrelevant attribute to zero (Tz is a tensor filled with zeros that is

Algorithm 1 DynamicPruning - Dynamic Weights For Layer Connections Using Text Support

```
1:  function CUSTOMMODULE
2:      Set BaseLinear($No$,$Ng$)
3:      Set DependantLinear($No$,$Ns$)
4:  end function
5:  function FORWARD(I, $Sc$, $Sa$)  ▷ Where I - input image, $Sc$ - supporting general text, $Sa$ - supporting specific
6:      $La$ = RetrieveRelevantParameters($Sc$)  ▷ Retrieve list of relevant parameter indices based on supporting general text
7:      for $k$ = 0 to $Na$ do
8:          if $k$ not in $La$ then
9:              DependantLinear.weight.data[k] = $Tz$
10:         else if $Sa$ != $-1$ and $Sa$ == $k$ then
11:             DependantLinear.weight.data[k] = $T\alpha$
12:         else
13:             DependantLinear.weight.data[k] = $T\beta$
14:         end if
15:     end for
16: end function
```

used to deactivate these connections). Line 13 presents the case where the range of possible attributes is adjusted. For example, all heel type attribute values are adjusted with a certain value β upon the detection of shoes category. In our text analysis, we get lots of hints about the available items (category, sub-category, attributes). We experimented using one/two hints from the detected attributes as a special case where the weight connections of these values are adjusted with value α (Line 11). For example, from the analysis of hashtags we get information about material such as: woolcoat. This is done for the purpose of seeing the effect of text analysis on enhancing the scope of classifications. In Algorithm 1, the categories layer is referred to as the base layer and the attributes layer is the dependant layer. Sc is the supporting category (e.g.: jeans) and Sa is the supporting attribute (e.g.: denim). **Figure** 2 (b) shows an example where Jeans category has been detected.

2) Multi-class Multi-labeled Scenarios: Likewise, for the multi-class multi-labeled scenarios, each image can be annotated with more than one category, and more than one sub-category and attributes. For this multi-category case, we still apply the same procedure we described in Algorithm 1, except that we consider more than one category while deciding the possible range of attributes or sub-categories. For example, with a skirt, and top detected, then the union of their sub-categories connections are adjusted, while the other connections are deactivated. The scope in such case is not defined in the same restricted way as for the single-class scenarios, due to the availability of more than one category. We noted that the Dynamic Layers model that we will describe briefly behaves better for multi-class scenarios.

Dynamic CNN With Text Guidance: With the possibility

Algorithm 2 DynamicLayers- Connecting with Different Layers Dynamically Using Text Support

```
function CUSTOMMODULE
    Set BaseLinear($No$,$Ng$)
    for i = 0 to $Ng$ do
        Set DependantLinear($No$,$La$[i])
    end for
end function
function FORWARD(I, $Sc$, $Sa$)   ▷ Where I - input image, $Sc$ - supporting general text, $Sa$ - supporting specific
    Decide DependantLinear based on $Sc$
    for $k$ = 0 to $Ai$ do
        if $Sa$ != $-1$ and $Sa$ == $k$ then
            DependantLinear.weight.data[k] = $T\alpha$
        else
            DependantLinear.weight.data[k] = $T\beta$
        end if
    end for
    MapCustomizedTensor(outputTensor, $Lai$)   ▷ Map output customized attributes tensor to unified attributes tensor for loss prediction
end function
function MAPCUSTOMIZEDTENSOR(OutputTensor, $Lai$)
    Set index = 0
    Initialize mappedTensor to a tensor of $Ng$ size
    for param in $Lai$ do
        mappedTensor[param] = OutputTensor[index]
        index += 1
    end for   ▷ This function sets the specific indices of parameters in the original tensor
end function
```

of having a dynamically computation graph for varying inputs, we present our second model in which layers can be chosen dynamically and become connected to the previous layer. Continuing on our example, for single-class multi-labeled scenario we define the following layers: (a) category linear layer that produces $\hat{C}$ of size $k = K$ representing the scores of predicted categories, (b) a set of attributes linear layers each producing $\hat{A}$ of size $p \leq P$ presenting the confidence scores of attributes predictions. For example: a layer containing attributes of dresses category. For multi-class multi-labeled scneario, in addition to the category layer we have: a set of subcategory linear layers each producing $\hat{S}$ of size $m \leq M$ that presents the confidence scores of subcategories, and an attributes linear layer producing $\hat{A}$ of size $p \leq P$. **Figure** 3 illustrates an example of multiple subcategory layers.

3) Single-class Multi-labeled Scenarios: For the single-class multi-labeled scenarios, based on the detected category, the specialized attributes layer is connected. For example, if the detected category is a dress, and hence the layer that contains the possible range of attributes for dresses is getting connected at run time. The followed algorithm is illustrated in **Algorithm** 2. Lines 3 - 5 in Algorithm 2 describe the definition of multiple dependant layers in the model (in this case: attributes layers). Line 8 in Algorithm 2 is responsible of deciding the suitable dependant layer to be connected based on the detected category. Lines 11 and 13 perform similarly as described before in Algorithm 1, to adjust the weights of the space of parameters based on supporting text. The output of the dynamic layer is further processed as we need to map it to a unified size to do the loss computations (function MapCustomizedAttributesTensor - line 16). To elaborate, assuming that 4 subcategory layers were added and processed during training, their output is mapped to one unified subcategory tensor that contains the values of all subcategories. Loss computation is done on the final tensor that got updated from the different participating subcategories.

4) Multi-class Multi-labeled Scenarios: For the multi-class multi-labeled scenarios, we add more than one sub-categories layer dynamically. For example, in **Figure** 3 the image is annotated with more than one category: skirt, tops, and accessories. This results in adding more than one sub-category layers dynamically. As previously mentioned, each category has a different set of sub-categories. The question mark in **Figure** 3 is a single-addition process for single-class scenarios and a multi-addition process for multi-class scenarios. In our approach, we use one category value and another sub-category value as supporting values during the classification. Both values are chosen randomly. After processing is done, the output of each of the newly added layers is mapped to the unified output size.

IV. EXPERIMENTS

Our architectures were implemented using Pytorch, and executed using a High Performance Tesla V100 GPU - 16 GB Memory in FloydHub platfrom[6].We run each experiment for 20 epochs, with initial learning rate (0.1) ,momentum (0.9). Values of α and β that are mentioned in the algorithms description were set to 0.1 and 0.01 respectively.

A. Datasets

The experiments were performed on two large datasets. The first dataset is a subset of the DeepFashion dataset [7], specifically from the category and attribute prediction benchmark[7]. The employed subset contains **10K** fashion images, annotated with 50 clothing category and 1000 attributes. We gathered a dataset consisting of Instagram posts from a community of fashion models (RewardStyle) [8]. The data is in the form of images, image captions, user comments, and hashtags associated with each post. The data was collected from public Instagram accounts. The dataset consists of 143 accounts, 200K posts, 9M comments. We applied the text analysis procedure which was described in **Section 3.1**. This resulted in annotating each image with 4 possible categories with percentages according to the importance, 4 sub-categories, 4 materials, 4 patterns, and 4 style options. We employed a subset consisting of **10K** images from the collected Instagram dataset, each with the described

[6]https://www.floydhub.com/settings/powerups

[7]http://mmlab.ie.cuhk.edu.hk/projects/DeepFashion/AttributePrediction.html

[8]https://www.rewardstyle.com/

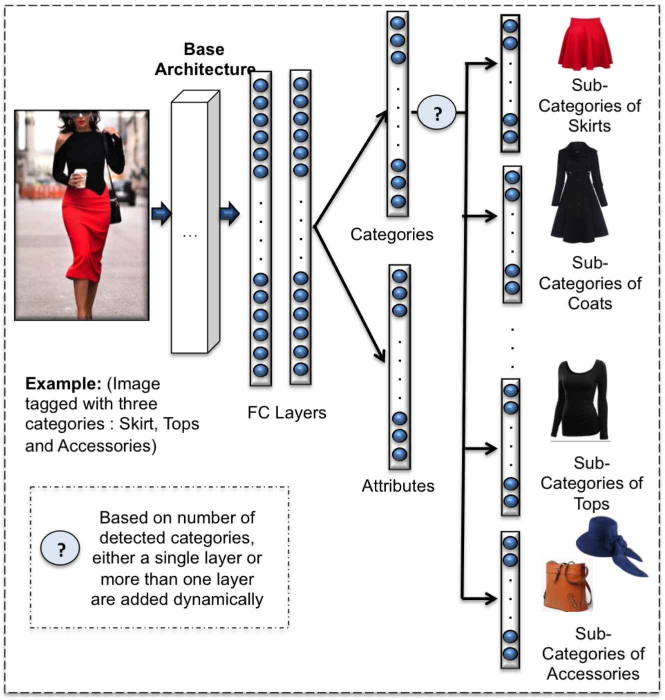

Base Architecture
...
FC Layers
Categories
Attributes
?
Sub-Categories of Skirts
Sub-Categories of Coats
Sub-Categories of Tops
Sub-Categories of Accessories
Example: (Image tagged with three categories : Skirt, Tops and Accessories)
Based on number of detected categories, either a single layer or more than one layer are added dynamically

Fig. 3. DynamicLayers architecture. Based on some of the detected text tag(s), one or more dynamic connections are created with the relevant sub-categories layer. Each sub-category layer has a different number of neurons.

annotations. In the coming two sections, we present the results of applying our algorithms in the two scenarios: single-class and multi-class multi-labeled images. We partition 80% of the data for training and 20% for validation.

B. Single-Class Multi-Labeled Classification Benchmark

For scenarios of single-class multi-labeled images classification, we have experimented on the DeepFashion [7] dataset. Each image in the mentioned dataset is tagged with 1 category and 2/3 attributes. We have selected two base CNN architectures to compare our models with. The first one is VGG16 [18], where we have replaced the last classification layers with 2 FC layers, and 1 layer for categories prediction and 1 layer for attributes prediction. We refer to this architecture as (BaseVGG16). Then, we have applied our Dynamic Pruning algorithm on this base VGG16 architecture, we refer to this architecture as (DynamicPruning16). Same applies for Dynamic Layers algorithm, where we have applied its structure on the base VGG16 architecture, referred to (DynamicLayers16). The other architecture that we have used in our experiments is ResNet [19]. We have replaced the last linear layers in the same way we did for VGG16. We refer to this architecture as (BaseResNet). Adaptive Neural Pruning and Dynamic Layers algorithms were also applied on the base ResNet archiecture. We refer to them as (DynamicPruningRes and DynamicLayersRes). For both DynamicPruning and DynamicLayers architectures, we need to decide the possible range of attributes per category which is not provided in the baseline dataset. To decide it, we have followed this procedure: we search for all images tagged with a certain category, for example: images tagged with dresses. Then, we group all attributes from all retrieved search results and we use them in the related layer as attributes belonging to that category. This dataset doesn't have text analysis information associated with images, so for the sake of experiment, we use the category from ground-truth as a supporting classification factor in a random way.

We have calculated the average precision and recall for the top 1, 3 and 5 multi-attributes classification according to the following rule: precision = $\frac{1}{n}\Sigma_n \frac{|(Yi \cap h(xi))|}{|(h(xi)|}$ and recall = $\frac{1}{n}\Sigma_n \frac{|(Yi \cap h(xi))|}{|Yi|}$ where n is the number of training images, Yi is the ground truth label assignments of the ith example, xi is the ith example, h(xi) is the predicted labels for the ith example. Precision for the detected attributes is the ratio of how much of the predicted attributes is correct. Recall is the ratio of how many of the actual attribute labels were predicted. **Table II** presents the **gain/loss** in average precision and recall values when comparing our models to the base architectures. As can be seen from the table, DynamicLayers and DynamicPruning architectures performed much better than the base architecture. For example, the average precision for Top 5 was increased with the value 21.2% by DynamicPruning16 over the base architecture (BaseVGG16). Another notable result is that there is no big difference in the behavior between our models for VGG16 architecture. Whereas DynamicLayers had better precision values in ResNet experiment than DynamicPruning. However, both models performed better than the base ResNet architecture. The values in the last two rows in **Table II** confirm that when comparing DynamicPruning and DynamicLayers, they performed similarly for VGG16, but DynamicLayers performed better with ResNet architecture. This can be probably related to the size of the network. **Figures** 4 and 5 graphically clarify the possible effect of the network size on the behavior of DynamicLayers algorithm. **Figure** 4 illustrates the similarity in values between DynamicPruning and DynamicLayers for VGG16, and the change in values is clearly presented in **Figure** 5 for ResNet as a base archiecture.

Interestingly, the average validation time per batch in seconds (**Figure** 6) was higher for DynamicLayers, DynamicPruing than our base architectures. This can be explained due to the additional processing that happens for customizing the connections. However, DynamicLayers**Res** and DymanicPruning**Res** average time was less when compared with DynamicLayers**16** and DynamicPruning**16**. These findings again encourage us to experiment with different network sizes to see their effect on the dynamic model. For average loss comparisons (**Figure** 7), DynamicLayers and DynamicPruning loss value was higher than the base architectures. This was expected as the network is changing its behavior and it needs time to adapt itself to the different patterns of data. We expect that the model's loss will be minimized with more training. As demonstrated in **Figure** 7, DynamicLayersRes and DynamicPruningRes was similar, whereas DynamicLayers16 was higher than DynamicPruning16. As expected, it might be

TABLE II
COMPARISON OF GAIN/LOSS ON AVERAGE PRECISION AND RECALL ON ATTRIBUTES PREDICTION (TOP 1, TOP 3 AND TOP 5)

	Average Precision			Average Recall		
	Top 1	**Top 3**	**Top 5**	**Top 1**	**Top 3**	**Top 5**
DynamicPruning16 - BaseVGG16	$\uparrow$ 15%	$\uparrow$ 21.1%	$\uparrow$ 21.2%	$\uparrow$ 28%	$\uparrow$ 37.9%	$\uparrow$ 38.2%
DynamicLayers16 - BaseVGG16	$\uparrow$ 15.4%	$\uparrow$ 21.1%	$\uparrow$ 21.1%	$\uparrow$ 27.9%	$\uparrow$ 37.9%	$\uparrow$ 38.2%
DynamicPruningRes - BaseResNET	$\uparrow$ 7.8%	$\uparrow$ 10.8%	$\uparrow$ 10.8%	$\uparrow$ 28.2%	$\uparrow$ 38.7%	$\uparrow$ 38.7%
DynamicLayersRes - BaseResNET	$\uparrow$ 15.5%	$\uparrow$ 21.4%	$\uparrow$ 21.4%	$\uparrow$ 28.1%	$\uparrow$ 38.4%	$\uparrow$ 38.4%
DynamicLayers16 - DynamicPruning16	$\uparrow$ 0.00045	$\downarrow$ 0.1338	$\uparrow$ 0.1572	$\uparrow$ 0.00355	$\downarrow$ 0.23945	$\uparrow$ 0.2259
DynamicLayersRes - DynamicPruningRes	$\uparrow$ 7.75365	$\uparrow$ 10.62285	$\uparrow$ 10.5991	$\downarrow$ 0.01605	$\downarrow$ 0.318954	$\downarrow$ 0.3981

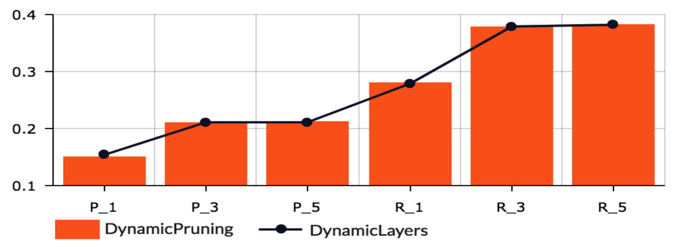


Fig. 4. Graphical illustration of the similar performance achieved by our algorithms when using VGG16 as a base architecture

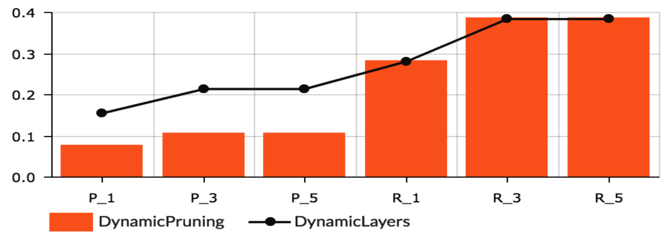


Fig. 5. Graphical illustration of the possible effect of network size on DynamicLayers algorithm. The chart shows difference in average precision values achieved by our algorihtms when using ResNet

attributed to the network size.

C. Multi-Class Multi-Labeled Classification Benchmark

We performed this benchmark using our Instagram dataset that has multi-class multi-labeled data. DeepPruning and DeepLayers algorithms were applied on the sub-categories layer for the interest of evaluating the hierarchical type of classifications. As explained in the DynamicLayers algorithm, we add more than one sub-categories layer dynamically, and then we calculate the precision and recall for the multi-class multi-labeled training examples (Equation 2 is modified by taking the sum over all classes). In Deep pruning, we keep modifying the same layer. For example, if a dress and skirt were detected, we modify the parameter space to accommodate to the attributes of the related categories. **Figure** 9 reveals that

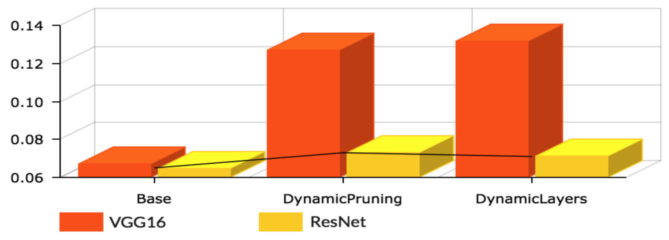


Fig. 6. Comparison of average validation time between the architectures under study.

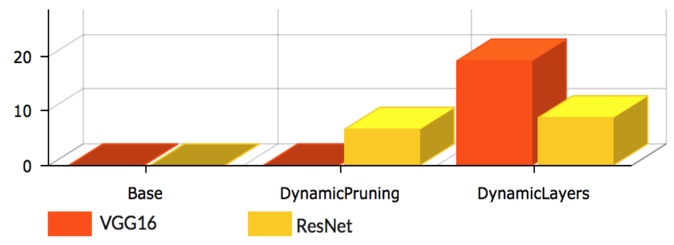


Fig. 7. Comparison of average loss values between the architectures under study.

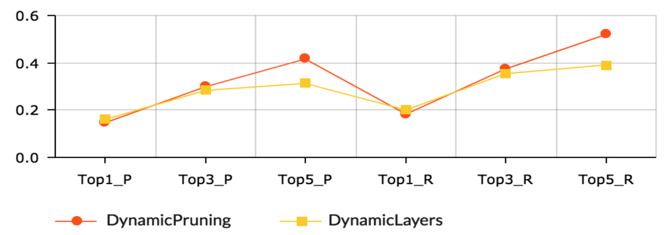


Fig. 8. Comparison of DynamicPruning and DynamicLayers average precision and recall for the categories layer.

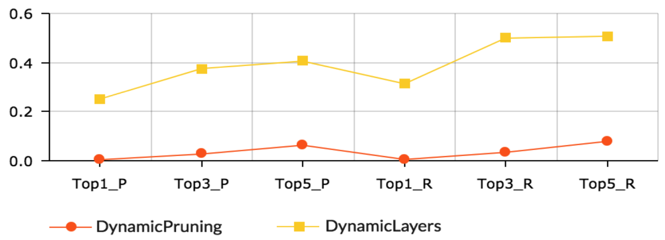


Fig. 9. Comparison of DynamicPruning and DynamicLayers average precision and recall for the sub-categories layer.

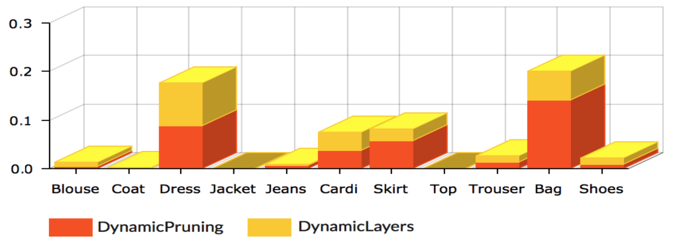


Fig. 10. Comparison of top 5 average precision for **category** classes prediction.

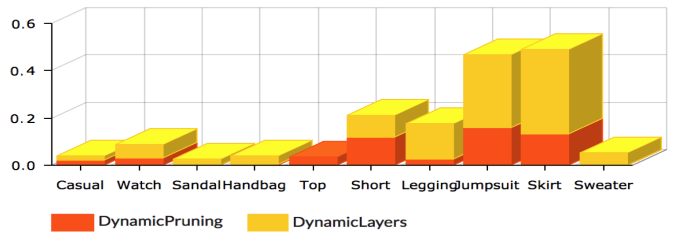


Fig. 11. Comparison of top 5 average precision for **sub-category** classes prediction.

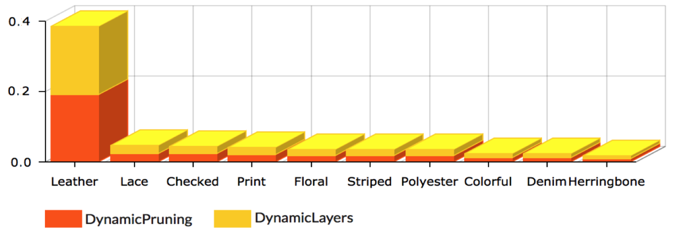


Fig. 12. Comparison of top 5 average precision for sample of **attribute** classes prediction.

DynamicLayers performed much better than DynamicPruning in sub-categories prediction. Specifically, DynamicLayers improved over DynamicPruning with a percentage of about %35 for average top 5 precision. This can be explained because with Dynamiclayers, the prediction range is more focused in each added layer, whereas in DynamicPruning, we activate the possible connections for more than one category, which widens the scope of the predictions. To elaborate, with DyanmicLayers, the addition of more than one layer and then calculating the final output caused better performance than DyanmicPruning where all the modifications happen in the same layer. This clearly explains the possible added value of using DynamicLayers for multi-class classifications. To compare the behavior of DynamicLayers and DynamicPruning, we have also calculated the average top 5 precision per category, subcategory, and attributes classes (**Figures** 10, 11, 12). In the figures, TopP refers to top precision and TopR refers to top recall values. **Figure** 10 presents a comparison on a category level of retrieval precision. The results indicate that both algorithms behaved similarly for retrieval per category class. Likewise, we compared the retrieval for specific sub-categories classes (for example: casual dresses, watches, handbags) (**Figure** 11), and for a sample of attributes classes (for example: leather, lace, floral) in **Figure** 12. It was also noted that both algorithms behaved similarly. This analysis was made on samples from sub-categories and attributes. The major categories in our Instagram dataset were dresses and skirts.

V. Conclusions and Experiments Summary

We propose for two novel techniques to incorporate social media textual content to support the visual classification in a dynamic way. Adaptive neural pruning and Dynamic Layers models are dynamic frameworks in which multiple classification layers exist and connections are activated dynamically based upon the mined text from the image. Our experiments show the improvments achieved by our models DynamicPruning and Dynamiclayers on the average precision and recall of image classification. The network size might be a factor that affects DynamicLayers algorithm. We have noticed that DynamicLayers and DynamicPruning algorithms behaved similarily for VGG16, but DynamicLayers behaved better for ResNet archiecture. The average validation time per batch was higher for DynamicLayers and DynamicPruning when compared to the base architectures. We assume that this is related to the processing that happens to choose the dynamic ranges. However, we plan to analyze the network's behavior for more than 20 epochs to see the effect of training time on reducing the validation time. The network might need more time to adapt to the patterns of data and changes in prediction scopes. The same explanation applies for the average loss. Interestingly, average loss was less for Resnet architectures when compared to VGG16.

References

[1] M. Schneier. (2014) Fashion in the age of instagram. [Online]. Available: https://www.nytimes.com/2014/04/10/fashion/fashion-in-the-age-of-instagram.html
[2] D.-A. Adegeest. (2016, May) Image power: fashion in the age of instagram. [Online]. Available: https://fashionunited.uk/news/fashion/image-power-fashion-in-the-age-of-instagram/2016051920461
[3] PromptCloud. (2019, Mar.) How instagram leverages ai and big data. [Online]. Available: https://medium.com/@promptcloud/how-instagram-leverages-ai-and-big-data-5f4ba7f035be
[4] S. Jaradat, "Deep cross-domain fashion recommendation," in *Proceedings of the 11th ACM Recsys*. ACM, 2017, pp. 407–410.
[5] C. E. Perez. (2017, Jan.) Pytorch, dynamic computational graphs and modular deep learning. [Online]. Available: https://medium.com/intuitionmachine/pytorch-dynamic-computational-graphs-and-modular-deep-learning
[6] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," 2016.
[7] S. Q. X. W. Ziwei Liu, Ping Luo and X. Tang, "Deepfashion: Powering robust clothes recognition and retrieval with rich annotations," in *Proceedings of IEEE CVPR*, 2016.
[8] Y.-N. Chen and H.-T. Lin, "Feature-aware label space dimension reduction for multi-label classification," in *NIPS*, 2012, pp. 1529–1537.
[9] H.-F. Yu, P. Jain, P. Kar, and I. Dhillon, "Large-scale multi-label learning with missing labels," in *International conference on machine learning*, 2014, pp. 593–601.
[10] L. Sun, S. Ji, and J. Ye, "Canonical correlation analysis for multilabel classification: A least-squares formulation, extensions, and analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 194–200, 2011.
[11] Y. Li, Y. Song, and J. Luo, "Improving pairwise ranking for multi-label image classification," in *The IEEE CVPR*, 2017.
[12] Y. Gong, Y. Jia, T. Leung, A. Toshev, and S. Ioffe, "Deep convolutional ranking for multilabel image annotation," *arXiv preprint arXiv:1312.4894*, 2013.
[13] F. Benites and E. Sapozhnikova, "Improving scalability of art neural networks," *Neurocomputing*, vol. 230, pp. 219–229, 2017.
[14] M. G. Augasta and T. Kathirvalavakumar, "Pruning algorithms of neural networksa comparative study," *Central European Journal of Computer Science*, vol. 3, no. 3, pp. 105–115, 2013.
[15] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *NIPS*, 2015, pp. 1135–1143.
[16] S. Anwar, K. Hwang, and W. Sung, "Structured pruning of deep convolutional neural networks," *ACM JETC*, vol. 13, no. 3, p. 32, 2017.
[17] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013, pp. 3111–3119.
[18] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE CVPR*, 2016, pp. 770–778.