Auto-Summarization: A Step Towards Unsupervised Learning Of a Submodular Mixture

Chandrashekhar Lavania*† Jeff Bilmes *‡

Abstract

We introduce an approach that requires the specification of only a handful of hyperparameters to determine a mixture of submodular functions for use in data science applications. Two techniques, applied in succession, are used to achieve this. The first involves training an autoencoder neural network constrainedly so that the bottleneck features have the following characteristic: the larger a feature's value, the more an input sample should have an automatically learnt property. This is analogous to bag of-words features, but where the "words" are learnt automatically. The second technique instantiates a mixture of submodular functions, each of which consists of a concave composed with a modular function comprised of the learnt neural network features. We introduce a mixture weight learning approach that does not (as is common) directly utilize supervised summary information. Instead, it optimizes a set of meta-objectives each of which corresponds to a likely necessary condition on what constitutes a good summarization objective. While hyperparameter optimization is often the bane of unsupervised methods, our approach reduces the learning of a summarization function (which most generally involves learning 2^n parameters) down to the problem of selecting only a handful of hyperparameters. Empirical results on three very different modalities of data (i.e., image, text, and machine learning training data) show that our method produces functions that perform significantly better than a variety of unsupervised baseline methods.

1 Introduction

Massive highly redundant sources of audio, video, speech, text documents, and sensor data have become commonplace and are expected to become larger and more preponderant in the future [58, 9]. Since these data sources are often redundant (e.g., surveillance video consists of long stretches of almost unchanging scenes interrupted occasionally by motion triggered false positives), processing them is inefficient and requires more compute and memory resources than necessary. Sometimes the size of a data set alone can make it infeasible to fully process, especially for expensive computational tasks such as multi-epoch deep neural network training.

One way to handle such large data sources is to reduce their size by summarizing them. Summarization typically requires optimizing an objective function that measures the quality and representativeness of any subset of the data. With such an objective, summarization becomes the problem of finding a small set having a high objective value. Finding and then optimizing this objective is a challenge, however, and thus a variety of methods have been used for this purpose.

One promising strategy restricts the summarization objective to be within a sub-class of functions, for example that of monotone non-decreasing submodular functions. Indeed, these are set functions that naturally model notions of diversity, dispersion, and coverage. Such functions have shown their merit in several summarization scenarios in different domains [27, 34, 57, 45, 26, 55, 50, 54]. Let $V = \{v_1, v_2, \ldots, v_n\}$, denote a ground set. Then a set function $f : 2^V \to \mathbb{R}$ is *submodular* if $f(a|A) \geq f(a|B)\ \ \forall A \subseteq B \subseteq V,\ a \in V \setminus B$, where $f(a|A) \triangleq f(a \cup A) - f(A)$ is the *gain* of adding element a to A. A set function is *monotone* if $f(A) \leq f(B)$ whenever $A \subseteq B$. A function f is *supermodular* if $-f$ is submodular, *modular* if it is both, and *normalized* if $f(\emptyset) = 0$. Typical summarization tasks maximize the objective subject to a given constraint [14, 28] leading to $\max_{A \subseteq V : |A| \leq k} f(A)$ for a cardinality constraint, a problem that can be solved approximately and efficiently using a simple greedy algorithm [35, 31, 33].

The set of monotone submodular functions is still vast, however, since for functions of the form $f : 2^V \to \mathbb{R}$ there are $O(2^{|V|})$ free parameters, giving a large search space within which to find a good objective. One approach is simply to construct a given f analytically based on high-level domain knowledge of the data source. A strategy that works still better attempts to learn the submodular function, and there are a variety of ways to do this. One common approach is to restrict the class still further to those that could be called "feature

*Department Of Electrical and Computer Engineering, University Of Washington, Seattle. 98195

†lavaniac@uw.edu

‡bilmes@uw.edu

based" functions[19], and are defined as

$$F_w(A) = \sum_{u \in U} w_u \phi_u(m_u(A)), \tag{1.1}$$

where U is a set of features, $w \in \mathbb{R}_+^U$ (for $u \in U$, $w_u \geq 0$ is a feature weight), ϕ_u is monotone non-decreasing concave function, and $m_u : V \to \mathbb{R}_+$ is a non-negative normalized modular function specific to feature u. For data item $v \in V$ and feature $u \in U$, $m_u(v)$ measures the degree of "u-ness" of v, and for $A \subset V$, $m_u(A) = \sum_{a \in A} m_u(a)$ is an additive measure of u-ness of the data items within A. Feature based functions have shown their usefulness in a variety of summarization tasks [37, 19, 54, 50]. They assume that data items can be represented as a non-negative, length-$|U|$ vector in feature space U. This is useful since the evaluation $f(A)$ of $A \subset V$ does not require access to all of V. Thus, these functions have a lower computational cost compared to functions whose evaluation requires access to the whole data (such as the $\mathrm{O}(n^2)$ cost required to query a facility location function [29]).

Fixing the set of modular functions $\{m_u\}_{u \in U}$, a variety of efforts [28, 59] have shown that it is possible to efficiently and effectively learn the $O(|U|)$ mixture weights of a feature based function in a large-margin fashion. The results of such learning typically produce summarization objectives that work much better than producing such a function by hand. Previous methods to learn the mixture weights have required supervised training data. Often this data takes the form of a set of pairs $\{A_i, f(A_i)\}_i$ for $A_i \subseteq V$, or triples $\{V_i, A_i, f(A_i)\}_i$ where $A_i \subseteq V_i$ and V_i is the i^{th} ground set. A still more limited form of supervised learning uses data of the form $\{V_i, A_i\}_i$ where V_i is a ground set and A_i is one possible high quality summary of V_i. In other settings, it is possible to learn the weights adaptively using various forms of online and/or bandit learning [60, 5]. In either case, it is necessary to obtain feedback in some form from the function being learnt.

The problem with the above approaches is that it is in general very hard to get any form of supervised training data. In an ideal world, it would be possible to train the weights in a fully unsupervised setting, with no (not even limited, indirect, or contextual) access to a true objective f. It is important to understand the importance of this problem. The fundamental problem is that there are many practical domains for which no example summaries or objective valuations are available from which to learn a good summarization objective in a supervised fashion. One example is massive scale image summarization, where V_i is a large (e.g., $n > 100k$) collection of images and we wish to produce summarizes of size 1000. Another example comes from environmental sensors (e.g., video, audio, temperature, humidity, air quality, EM radiation) and smart networks. Smart cities are being instrumented with an ever increasing variety and amount of such sensors which produce torrents of data that are likely highly redundant, and thus could benefit from summarization. Problematically, there is no way to train a summarization objective since ground-truth summaries are unavailable; there are no pre-existing known-to-be-representative subsets from which to learn a summarization objective (not to mention the existence of such subsets should be the result, rather than the input, of a summarization process). Acquiring such training data is unlikely to occur since the labeling task (i.e., producing summarizes of very large data sets) is neither natural nor feasible for humans, unlike other machine learning problems.

In the present paper, we take a step towards an unsupervised approach to learning the mixture weights. While the learning of all $O(2^{|V|})$ free parameters is hard [1, 13], and learning even the mixture weights would require a large amount of training data, our approach is to reduce the problem down to finding a good setting for a handful of hyperparameters. The hyperparameters determine a meta-optimization that is used to determine the mixture weights, and since there are relatively so few hyperparameters, it becomes feasible to select them with only a very small amount of evaluation information. Thus, while our approach is not fully unsupervised, the paper, as far as the authors have been able to discern, proposes the first solid advancement towards the unsupervised learning of submodular mixtures since perhaps [56] who instantiated a function using Gaussian mixture models and HMMs which were themselves produced using unsupervised learning. Given as input a desired number $|U|$ of features, and settings for the hyperparameters, our method learns the features U, the modular functions $\{m_u\}_{u \in U}$ where $m_u : V \to \mathbb{R}_+$ in an unsupervised manner, and also learns the mixture weights for a feature based function using methods described below.

2 Learning "AC" Features and Modularity

In a deep neural network based autoencoder, we wish to construct two mappings, an encoder $\mathfrak{e} : \mathscr{D} \to \mathbb{R}^d$ and a decoder $\mathfrak{d} : \mathbb{R}^d \to \mathscr{D}$, where $\mathscr{D}$ is the domain of x. To build modular functions using such mappings, we define $U = \{1, 2, \ldots, d\}$ so that the features correspond to the elements in the encoded data items. If $v \in V$ is an index of a data item, x_v is the corresponding data item, then we would set $m_u(v) = \mathfrak{e}(x_v)(u)$ i.e., the u^{th} element in the vector encoding $\mathfrak{e}(x_v)$ of input x_v. To ensure the feature functions are submodular, we need $m_u(v) \geq 0$ for all u and v. Fortunately, this is easy attained since we can use an autoencoder where the non-linearity (such as logistic, or ReLU) at the bottleneck is never negative. Hence, if $A \subseteq V$ is a set of input items, then we set, for all $u \in U$, $m_u(A) = \sum_{a \in A} \mathfrak{e}(x_a)(u)$.

There is a potential problem with such learnt representations, however, in that they might not have a necessary characteristic for them to be useful in feature based functions (Eq. (1.1)), even if the features are non-negative. In Eq. (1.1), a higher value of $m_u(v)$ should imply a higher presence of property u in data sample v. While with an autoencoder, the meaning of the feature u is automatically learned, it is not necessarily the case that as $\mathfrak{e}(x_v)(u)$ gets larger, the u property correspondingly gains in prominence. For example, given two input samples x_{v_1} and x_{v_2}, if it is the case that $\mathfrak{e}(x_{v_1})(u) < \mathfrak{e}(x_{v_2})(u)$, then we wish for object v_2 to have more of whatever property is represented by u than object v_1. This characteristic is similar to "bag of words" vector representations, where each element of the vector is proportional to a count of how often a given term or n-gram has occurred in a document, a characteristic also true of most TFIDF [41] representations. Coincidentally, the very same characteristic that is useful for bag-of-words representations is what we need to be present in the bottleneck representation in an autoencoder. What we are striving for is a strategy to automatically learn a mapping from x to the non-negative orthant of a d-dimensional space where x's coordinate therein represents the degree to which each of $|U|$ learnt words are prevalent within x. Moreover, if we have two or more objects indexed by $A \subseteq V$, then the objects should contribute to the presence of property u additively, as in $\sum_{a \in A} \mathfrak{e}(x_a)(u)$. For this reason, we call this characteristic "*additively contributive*" (AC), and such features are called "AC features."

We next propose an unsupervised strategy to train a deep autoencoder to produce features that are likely to have an AC interpretation. Consider the encoding-decoding process, $\mathfrak{d}(\mathfrak{e}(x))$ of an object x. Normally, the first operation of the decoder, after the encoding of x, is a matrix multiply. Let $W \in \mathbb{R}^{d' \times d}$ be a $d' \times d$ matrix corresponding to the first matrix multiply in the decoder. Moreover, let $\mathfrak{d}'$ be the remainder of the decoding process after this matrix multiply. Hence we have that $\mathfrak{d}(\mathfrak{e}(x)) = \mathfrak{d}'(W\mathfrak{e}(x))$ where $W\mathfrak{e}(x)$ is a matrix-vector multiplication. Once we train the autoencoder, the matrix W can consist of positive or negative values. The negative values of W can lead to the encoded features not having an AC interpretation since as the non-negative quantity $\mathfrak{e}(x)(u)$ gets larger, if it is multiplied by a negative weight within W, the contribution to the final output could decrease rather than increase. One approach, which we adopt in the present work, therefore is to restrict W to be non-negative during the training process, as in $\min_{\mathfrak{d}',\mathfrak{e},W \in \mathbb{R}_+^{d' \times d}} E_{x \sim p}[\|x - \mathfrak{d}'(W\mathfrak{e}(x))\|]$ where we minimize the construction error subject to the matrix W being non-negative — we call this "AC constrained training." This optimization can be achieved using projected stochastic gradient descent, where after each gradient step, we project W back into the positive orthant (i.e., $W \in \mathbb{R}_+^{d' \times d}$) which is extremely easy to do exactly by truncating any negative values to zero.

Some points regarding AC constrained training are addressed next. It might be possible that the trained model converges to a point where W is close to an identity transform, essentially passing the decoding "buck" to the remaining layers within $\mathfrak{d}'$. One way to eliminate this possibility is to restrict the entire decoder to use only non-negative matrices (which would only be sensible with nonlinearities unlike ReLU since this would make the entire decoder linear). Alternatively, if during the training process, the matrix W tends closer to an identity-like matrix, we can explicitly discourage this using repulsive regularization [39, 30]. Yet another strategy is to use an asymmetric encoder-decoder, where the decoder uses fewer layers than the encoder, and hence in order for the decoder to be effective, training cannot afford to essentially bypass a layer by it acting as an identity transform. In our experimental results, we found that the simple strategy of projecting only W back to the positive orthant was sufficient and, while positive, W was never close to any form of identity transform. Moreover, summarization results (Section 5) using the resulting modular functions always demonstrated a significant benefit to AC constrained training over unconstrained training.

Given an AC-constrained trained autoencoder, we can then construct feature-based functions as in Eq. (1.1) where $m_u(A) = \sum_{a \in A} \mathfrak{e}(x_a)(u)$. A potential problem with this approach, however, is that the encoded features might not be well calibrated with the concave function within which they are used. To produce a good summarization objective, we need to be sure that the diminishing returns property of the concave function ϕ_u occurs commensurately with the typical range of values of $m_u(A)$. In order to achieve this, we introduce a set of calibration parameters that scale the encoder outputs to match their corresponding concave function, as in the following:

$$
\text{(2.2)} \quad F_w(A) = \sum_{u \in U} \sum_{\gamma \in \Gamma} w_{u,\gamma} \frac{\phi_u\left(\gamma \frac{\sum_{a \in A} \mathfrak{e}(x_a)(u)}{\sum_{v \in V} \mathfrak{e}(x_v)(u)}\right)}{\phi_u(\gamma)} = \sum_{u \in U} \sum_{\gamma \in \Gamma} w_{u,\gamma} \phi_{u,\gamma}(m_u(A)).
$$

Here, Γ is a fixed set of positive real valued calibration constants, and $w_{u,\gamma}$ is the weight corresponding to γ-calibrated feature u. Also, $\phi_{u,\gamma}(\alpha) = \phi_u(\gamma\alpha)/\phi_u(\gamma)$, and $m_u(A) = \frac{\sum_{a \in A} \mathfrak{e}(x_a)(u)}{\sum_{v \in V} \mathfrak{e}(x_v)(u)}$ is a modular function normalized so that $0 \leq m_u(A) \leq 1$ — the quantity $\sum_{v \in V} \mathfrak{e}(x_v)(u)$ is a normalizing constant to ensure that the argument to ϕ_u lies within the range $[0, \gamma]$.

The division by $\phi_u(\gamma)$ ensures that $\forall A \subseteq V$, $0 \leq \phi_{u,\gamma}(m_u(A)) = \phi_u\left(\gamma \frac{\sum_{a\in A} \mathfrak{e}(x_a)(u)}{\sum_{v\in V} \mathfrak{e}(x_v)(u)}\right)/\phi_u(\gamma) \leq 1$, thus allowing us w.l.o.g. to assume $0 \leq w_{u,\gamma} \leq 1$ and $\sum_{u\in U,\gamma\in\Gamma} w_{u,\gamma} = 1$. In the expression above, we consider ϕ_u a "mother function" and $\phi_{u,\gamma}$ a calibrated instantiation.

3 Towards Unsupervised Learning of Weights

Our next step is to learn the weights w of the mixture (Equation 2.2) with minimal hyperparameters and without the incorporation of ground truth information in the learning objective. We introduce a meta-objective $J(w)$ that measures the quality of the mixture weights, and our problem becomes:

$$\max_{w\geq 0, ||w||=1} J(w) \tag{3.3}$$

Since no supervised information is available, we design the objective $J(w)$ to measure a set of properties of $F_w(\cdot)$ that we believe to be necessary (more or less) for it to act as a good summarization objective for the given modular functions $\{m_u\}_u$ and this is done by setting $J(w)$ itself to be a mixture of meta-objectives, weighted by meta-hyperparameters, each one measuring one necessary condition. The following describes the set of necessary conditions.

3.1 J_1: Confidence A good summarization objective should be able to, with confidence, claim the top summary to be good. J_1 is therefore designed to, when maximized, ensure that there is an appreciable score gap between the top summaries and the bottom summaries, and this is done as follows:

$$J_1(w) = \mathbb{E}_{k\sim p}\left[\left|\max_{S\subseteq V, |S|=k} F_w(S) - \min_{S'\subseteq V, |S'|\geq k} F_w(S')\right|\right] \tag{3.4}$$

where p is a distribution over summary size values. Details on which p we use in practice are given in Section 5.

3.2 J_2: Entropy A feature function could have high confidence according to J_1 (Eq. (3.4)), even if only a small set of components in $F_w(\cdot)$ have significant weight, creating a biasing in favor of the small set, and reducing the ability of a feature based function to encourage diversity when maximized. To discourage this, J_2 encourages the entropy of the distribution w to remain high, leading to:

$$J_2(w) = -\sum_{u,\gamma} w_{u,\gamma} log(w_{u,\gamma}) \tag{3.5}$$

In other words, J_2 acts as a regularizer in the optimization of the overall objective J.

3.3 J_3: Non-modularity of Summary Scores The degree to which $F_w(\cdot)$ is modular corresponds to the degree to which the elements being summarized are independent. Our contention is that a good summarization objective should not be strongly modular, since a modular function being an appropriate summarization objective would imply that there is no redundancy at all in the original data, a property we have seldom observed. We measure this form of non-modularity via:

$$J_3(w) = \mathbb{E}_{k\sim p}\left[\sum_{s\in\hat{S}} F_w(s) - F_w(\hat{S}) \;\middle|\; \hat{S} \in \operatorname*{argmax}_{S\subseteq V:|S|\leq k} F_w(S)\right] \tag{3.6}$$

where again p is distribution over summary sizes (§5).

3.4 J_4: Curvature Another mechanism to measure closeness to modularity is the total curvature of the set function [6]. If the curvature is zero, the function is modular and as it deviates from zero, the function becomes more curved until it reaches maximum curvature at one. This is defined as:

$$J_4(w) = 1 - \min_{j\in V} \frac{F_w(j|V\setminus j)}{F_w(j)}. \tag{3.7}$$

Objectives $J_4(w)$ and $J_3(w)$ are related. However $J_4(w) = 1$, its maximum possible value, does not imply $J_3(w)$ is maximized. Hence, we include both.

3.5 J_5: Stability For any setting of w that produces a good summarization objective, small perturbations of w should not dramatically change the summaries. Analogous to the confidence measure above, we can measure this by discouraging a large change in summarization score under a small change in w. Let $\hat{w}$ be a random variable centered around w and governed by distribution $p_w(\hat{w})$. We can measure the stability of w as follows:

$$J_5(w) = -\mathbb{E}_{p_w}\left[\mathbb{E}_{k\sim p}\left[\max_{S\subseteq V,|S|=k} F_w(S) - \max_{S'\subseteq V,|S'|=k} F_{\hat{w}}(S')\right]^2\right]. \tag{3.8}$$

Maximizing J_5 discourages small perturbations to w from causing drastically different summaries. In practice, we use w-mean Gaussian noise followed by projection back onto the simplex [12].

3.6 J_6: Soft De-duplication before saturation Ideally, a summarization process should not include redundant elements into a partial summary S, where $|S| < k$, unless the objective f has reached saturation (i.e., when $f(S) \approx f(V)$). We introduce an *unsaturation degree* defined as $u(S) = 1 - F_w(S)/F_w(V) \in [0,1]$. When $u(S)$ is large, any new item v added to S should not be similar to any item in S. Define $\psi(v,S) = \max_{s \in S} \text{sim}(v,s)$ to be maximum similarity between v and any element in S. When $u(S)$ is large, therefore, $f(v|S)$ should be large whenever $\psi(v,S)$ is small. An objective to encourage this behavior is defined as:

$$J_6(w) = \mathbb{E}_{k \sim p}\left[\sum_{i=1}^{k}\left(1 - \frac{F_w(\hat{S}_i)}{F_w(V)}\right)\frac{F_w(v_i|\hat{S}_{i-1})}{\psi(v_i,\hat{S}_{i-1})} \middle| \hat{S}_i \in \text{gargmax}_{S \subseteq V, |S|=i} F_w(S), v_i \in \hat{S}_i \setminus \hat{S}_{i-1}\right] \tag{3.9}$$

Here, "gargmax" returns the ordered set computed via the greedy algorithm.

3.7 Combined Objective The combined objective takes a weighted mixture of the above six components,

$$J(w) = \sum_i \lambda_i J_i(w). \tag{3.10}$$

The set $\lambda \triangleq \{\lambda_i\}_i$ act as six meta-hyperparameters. When these are fixed, our method can learn the mixture w in an unsupervised manner since none of the objectives $J_i(w)$ need supervised feedback (in the form of information about any true objective) for training.

We thus consider this approach a strategy for reducing the number of hyperparameters. A summarization objective $f : 2^V \to \mathbb{R}_+$ may have 2^n free parameters, where $n = |V|$. Using only feature functions, we have as potential hyperparameters the modular functions m_u and the mixture weights w. Section 2 discussed how to first learn the modular functions in an unsupervised fashion. This leaves us with, at most, $|U||\Gamma|$ mixture weights to learn. Optimizing the mixture using $J(w)$ reduces this further to the six meta-hyperparameters in λ, a drastic reduction from 2^n. Hyperparameter tuning is itself an active area of research [52, 2] and a plethora of hyperparameters is challenging, particularly in the unsupervised learning setting. By significantly reducing the number of hyperparameters, as we have done, this issue is somewhat mitigated.

For a given fixed λ, the objective $J(w)$ is optimized using gradient ascent. For each update, the following steps are followed: (1) Choose a random size $k \sim p$; (2) perform discrete optimizations for those meta-objectives that need them (e.g., J_1, J_3, J_5, and J_6); (3) Compute the gradient $\nabla_w J(w)$ at the discrete solutions; (4) adjust w with a gradient step under a given learning rate (using Adagrad [11]); and (5) project w back to the probability simplex using [12]. The discrete optimizations in J_3, J_5, J_6, and the first part of J_1 can be near optimally solved, with a multiplicative approximation guarantee of $1 - 1/e$, using the greedy algorithm [35], since these all involve cardinality constrained monotone submodular maximization. The constrained minimization in the second part of J_1 can be approximately solved using the semidifferential methods of [17].

4 Previous Relevant Work

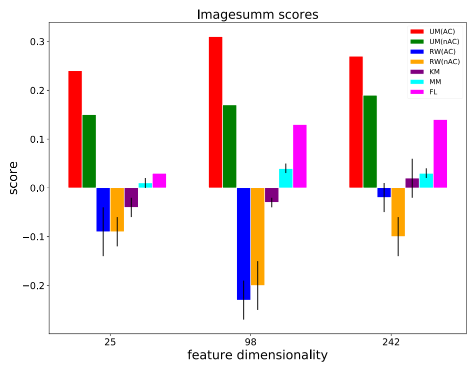
Features with constituent elements having the characteristic that a higher value of the element corresponds to a higher inclusion of the corresponding property in the data object have seen usage in past literature. One such case are the well known "bag of words" features. Bag-of-words features have been successfully used in many text processing [42], natural language processing [21] and computer vision [38] tasks. These are only a few examples and do not do justice to the vast literature in this area.

In the area of computer vision, features generated using deep neural networks (sometimes treated as a black box) have been successfully converted into the form of a bag of visual words. One such transformation was used by Tschiatschek et al. [50]. They use layer 17 of OverFeat [43] to generate intermediate features. These features are then clustered using k-means and encoded using a kernel code book to generate bag of words features. Our attempt, however, generates the additively contributive features (as described earlier) directly and more explicitly from the deep neural network. Our deep neural network generators are trained to produce such features without the need of extra processing as a subsequent step.

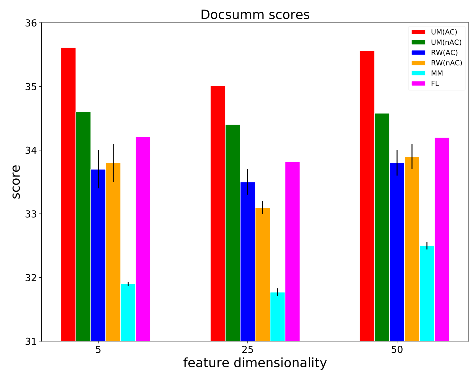
The process of constraining matrices is also an integral part of non-negative matrix factorization (NMF [24]). NMF involves the factorization of a matrix V into matrices W and H such that $V \approx WH$ with the constraint that each of these matrices have only non-negative elements. Clustering is an inherent part of NMF [7] allowing its use for applications that require grouping of data.

The benefits of constraining internal representations in a neural network have been explored by Trigeorgis et al. [49]. They propose a Deep semi-NMF (Semi Non Negative Matrix Factorization) model and show that a neural network with such constraints can learn representations of the data that are beneficial for clustering based on the attributes of the neural network layers. Unlike our work, however, they do not directly constrain the neural network weights.

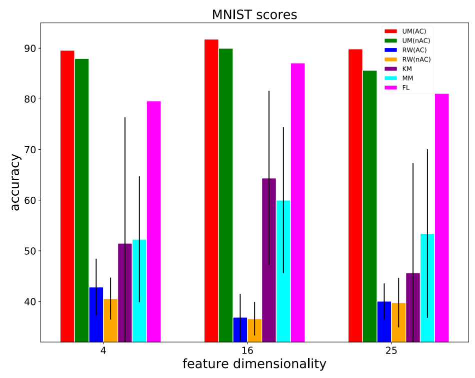
In the present work we study an approach towards unsupervised learning of submodular mixtures. As far as we know, there has been no prior attempt to learn such mixtures in an unsupervised manner. However, the fully supervised learning of these mixtures has been explored in the past. Sipos et al. [44] use large



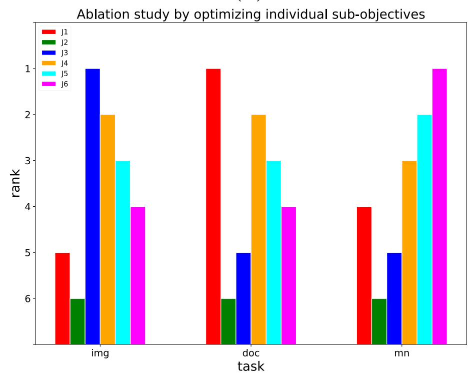
(a)



(b)



(c)



(d)

Figure 1: Comparison of Unsupervised Mixture (UM) for AC and non-AC (unconstrained) bottleneck features, random weights (RW), k-means (KM) (for cardinality constraint scenario), MMR-like baseline (MM) and a similarity based Facility Location (FL) submodular function. (a) Normalized VROUGE scores, (b) ROUGE-1 DUC-04 recall scores, (c) Classification performance (accuracy in %) for a model trained on a 1k subset of training data. (d) Ablation study depicting rank of the meta-objectives for different tasks(img: image summarization, doc: document summarization, and mn: subset selection). Rank 1 is best.

margin learning [51] to learn a monotone function for document summarization. Lin and Bilmes [28] were able to learn the mixtures weights in the context of document summarization using loss augmented inference [47]. Similarly, Tschiatschek et al. [50] applied the supervised mixture learning procedure to image summarization. Gygli et al. [14] extended its usage to video summarization.

5 Experiments

We show that our approach performs significantly better than several baseline unsupervised methods on three distinct summarization tasks: (1) image summarization, (2) textual document extractive summarization, and (3) training data summarization. We test both AC-constrainedly trained as well as unconstrained autoencoder based features for $\{m_u\}_u$, and find that the AC-constrained features always improve over their unconstrainedly trained counterparts. For each case, the different autoencoders have exactly the same structure, the only difference being in the constraint placed on W during training. Moreover, we find that our mixture learning approach via optimization of $J(w)$ leads to appreciably improved summarization objectives. Due to the lack of previous work on learning mixtures of submodular functions with no direct utilization of a ground truth summary in the learning objective, we compare our work with other methods such as k-means (for the cardinality constrained case) along with Query independent MMR (Maximal Marginal Relevance) [4], and the similarity based and resource intensive "Facility Location" (FL) submodular function [32, 53]. FL is submodular and MMR has a diminishing returns property and thus they provide strong baselines. We utilize Euclidean distance for similarity calculations in FL and MMR . These methods are either unsupervised or require minimal supervision for instantiation of the submodular function. [1]

[1]Additional details are available in the extended version of the manuscript

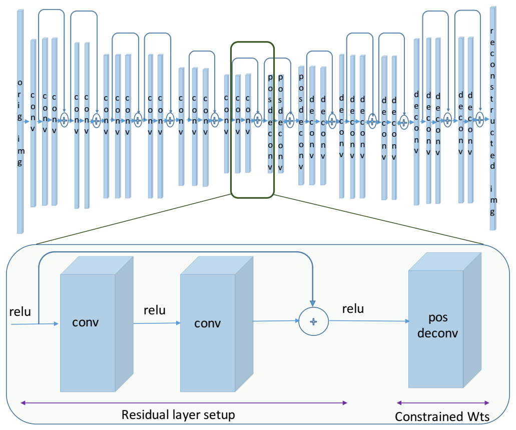


Figure 2: Sample autoencoder architecture for constructing AC image features. The zoomed view shows that the first layer after the bottleneck is pos deconv. A pos deconv layer's weight matrices are non-negatively constrained during training.

5.1 Image Summarization We generate AC autoencoder features from images and use them to instantiate $F_w(\cdot)$ to learn w. The features are generated by training image to image convolutional autoencoders. The network architectures are designed with the intention of preventing any bypassing of the bottleneck. Therefore, no pooling layers were used. Such layers typically need corresponding unpooling layers and these unpooling layers can allow transfer of information between the encoder and decoder parts of the network without passing through the bottleneck. In addition to this, our neural networks employed residual layers [15]. A residual layer was constructed using pairs of similar layers (either convolutional or deconvolutional). Figure 2 shows a sample architecture that was used to extract features from images. The residual block at the bottleneck is shown as part of the zoomed view. This view also depicts the first deconvolution layer in the decoder part of the architecture.

The bottleneck layer has direct connections to only "pos deconv", a deconvolutional/residual block whose weights are constrained to be non-negative during training. These weights were projected to the non-negative orthant after each back propagation update, achieved by setting all negative values to zero. The networks were trained using ADAM [18] with batch normalization and rectified linear units. The training was performed using the ILSVRC 2012 data set, part of the ImageNet Large Scale Visual Recognition challenge for object classification [40]. The data has around 1.28 million images in the training set and 50 thousand images in the validation set. Neural networks trained on this data set have been successfully used as feature extractors for a variety of tasks [8, 22].

We used a standard dataset for image summarization consisting of 14 sets each containing 100 images [50]. These sets have a list of associated user summaries. Each summary contains 10 images. Image summarization can be formulated as cardinality constrained submodular maximization, as in [50]. The set V corresponds to one of the sets of 100 images which is to be summarized. Given a budget $k = 10$, and the trained F_w, we find $S^* \in \text{argmax}_{S \subseteq V, |S| \leq k} f(S)$, easily approximable with the greedy algorithm [35]. Summaries are evaluated using the normalized VROUGE score as described in [50]. None of the ground truth summaries from [50] were used to train F_w, rather they were used only to evaluate the hypothesized summaries produced via the learned F_w.

We learned F_w as described in Section 2 and 3. During training, we set $\not{p} = \text{uniform}[1, 50]$. Fig 1a shows the normalized VROUGE (higher is better) results. We show results for various $|U| = d \in \{25, 98, 242\}$ and also compare with several submodular and non submodular baselines. The benefit of AC over nonAC can be observed. In addition to this Fig 1a demonstrates that our approach can outperform the baselines.

5.2 Extractive Document Summarization We next test on the 2003 & 2004 NIST Document Understanding Conference (DUC) [10] data sets which provide standard benchmarks for query independent multi-document extractive summarization. We perform training on DUC 03 and evaluate on DUC 04. Summaries consist of text less than 665 bytes long. The data was pre-processed using porter stemmer and the generated summaries are evaluated using the ROUGE-1 score [25] (Information about ROUGE-1 on DUC03 or DUC04 was not used at any time during training). Summarization itself is done using knapsack constrained [46, 16] optimization and the formulation is otherwise similar to the previous section. The greedy summarization procedure utilized the summary cost only to ascertain that the budget had not been breached.

The experiments required the generation of both AC-constrained and unconstrained features, and this was done via a two step process. First, we used data from DUC along with data from the NEWS2013 data set [36] to generate 500 dimensional sentence vectors. These sentence vectors were generated by training a LSTM based sequence to sequence network using OpenNMT [20]. The model used a two layer encoder along with a decoder that contained 500 dimensional hidden states. Next, the sentence vectors for DUC data were used to train both an AC and a non-AC constrained autoencoder, which were used to produce AC and non-AC features. These were then used to instantiate $F_w(\cdot)$. We set $\not{p} = \text{uniform}[1, M]$ where M is the size of the document under consideration for gradient calculation. The associated ROUGE-1 recall scores on DUC 04 are reported in Fig 1b. Again, we see a significant benefit of AC over non-AC features, as well as a significant benefit of our approach over the baselines.

5.3 Training Data Summarization As training a machine learning model can be expensive, we next test if our method can produce a summarization objective that is able to choose a good subset of training data. The evaluation is accuracy on a test set to see if the selected training set is a good training subset. We utilize the MNIST dataset [23] where the standard training set was first divided into a set containing 50,000 images (V_{train}) and a validation set containing 10,000 images (V_{val}). We produce both AC and non-AC features. The features were then used to instantiate $F_w(\cdot)$. During training, we set $\rlap{p}{\,\sim} \sim$ uniform$[1, 200]$. The experiments were conducted to produce a subset of $k = 1000$ images from V_{train}. Next, a ReLU-based LeNet [23] model was trained on the selected subset and validated on the validation set. Results are given in Fig 1c. Once again AC-feature based optimized $J(w)$ achieves the best results.

5.4 Ablation study for $J_i(\cdot)$ As an ablation study, we explore the performance of the individual $J_i(w)$ for the tasks defined earlier (using the largest bottleneck size). Figure 1d, displays the rank of each meta-objective. It can be observed that their importance is dependent on the type of data and task in consideration. Nevertheless, a rough pattern does emerge. J_4 (curvature) is consistently ranked high across tasks. Similarly, J_5 (stability) and J_6 (soft de-duplication) are the next most consistent performers. This lends credence to the importance of these meta-objectives and the overall submodular nature of the summarization task itself (high curvature consistently performs well).

6 Conclusions

An advancement towards unsupervised learning of submodular mixtures is proposed in this work. The approach is a two-part process. It involves unsupervised learning of features that are especially useful for submodular function instantiation. It also includes learning the mixture weights through several meta-objectives (that do not directly incorporate any ground truth information) combined together using meta-hyperparameters. It is empirically shown over three very different modalities, the proposed approach works significantly better than several comparable unsupervised baselines.

This work was supported by the CONIX Research Center, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

References

[1] M.-F. Balcan and N. J. Harvey. Learning submodular functions. In *ACM symposium on Theory of computing*, pages 793–802. ACM, 2011.

[2] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *JMLR*, 13(Feb):281–305, 2012.

[3] E. Cambria, S. Poria, D. Hazarika, and K. Kwok. Senticnet 5: discovering conceptual primitives for sentiment analysis by means of context embeddings. In *AAAI*, 2018.

[4] J. Carbonell and J. Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*. ACM, 1998.

[5] L. Chen, A. Krause, and A. Karbasi. Interactive submodular bandit. In *NIPS*, 2017.

[6] M. Conforti and G. Cornuejols. Submodular set functions, matroids and the greedy algorithm: tight worst-case bounds and some generalizations of the Rado-Edmonds theorem. *Discrete Appl. Math.*, 7(3):251–274, 1984.

[7] C. Ding, X. He, and H. D. Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *SDM*. SIAM, 2005.

[8] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014.

[9] A. Dube, A. Helkkula, et al. Customer approach to the use of big data: Wearables for service. In *SERVSIG*, 2016.

[10] The nist document understanding conference (duc) data, 2004.

[11] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12(Jul):2121–2159, 2011.

[12] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l1-ball for learning in high dimensions. In *ICML*. ACM, 2008.

[13] V. Feldman and J. Vondrák. Optimal bounds on approximation of submodular and XOS functions by juntas. *CoRR*, abs/1307.3301, 2013.

[14] M. Gygli, H. Grabner, and L. Van Gool. Video summarization by learning submodular mixtures of objectives. In *CVPR*, 2015.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[16] R. Iyer and J. Bilmes. Submodular optimization with submodular cover and submodular knapsack constraints. In *NIPS*, 2013.

[17] R. Iyer, S. Jegelka, and J. Bilmes. Fast semidifferential-based submodular function optimization. In *ICML*, 2013.

[18] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *preprint arXiv:1412.6980*, 2014.

[19] K. Kirchhoff and J. Bilmes. Submodularity for data selection in machine translation. In *EMNLP*, 2014.

[20] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. Opennmt: Open-source toolkit for neural machine translation. *preprint arXiv:1701.02810*, 2017.

[21] M. Lapata and F. Keller. The web as a baseline: Evaluating the performance of unsupervised web-based models for a range of nlp tasks. In *HLT-NAACL*, 2004.

[22] C. Lavania, S. Thulasidasan, A. LaMarca, J. Scofield, and J. Bilmes. A weakly supervised activity recognition framework for real-time synthetic biology laboratory assistance. In *Ubicomp*, pages 37–48. ACM, 2016.

[23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[24] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, 2001.

[25] C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8, 2004.

[26] H. Lin and J. Bilmes. Multi-document summarization via budgeted maximization of submodular functions. In *HLT-NAACL*, pages 912–920, 2010.

[27] H. Lin, J. Bilmes, and S. Xie. Graph-based submodular selection for extractive summarization. In *Proc. IEEE Automatic Speech Recognition and Understanding (ASRU)*, Merano, Italy, December 2009.

[28] H. Lin and J. A. Bilmes. Learning mixtures of submodular shells with application to document summarization. *preprint arXiv:1210.4871*, 2012.

[29] Y. Liu, K. Wei, K. Kirchhoff, Y. Song, and J. Bilmes. Submodular feature selection for high-dimensional acoustic score spaces. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 7184–7188. IEEE, 2013.

[30] J. Malkin and J. Bilmes. Ratio semi-definite classifiers. In *ICASSP*. IEEE, 2008.

[31] M. Minoux. Accelerated greedy algorithms for maximizing submodular set functions. *Optimization Techniques*, pages 234–243, 1978.

[32] P. Mirchandani and R. Francis. *Discrete location theory.* 1990.

[33] B. Mirzasoleiman, A. Badanidiyuru, A. Karbasi, J. Vondrák, and A. Krause. Lazier than lazy greedy. In *AAAI*, pages 1812–1818, 2015.

[34] B. Mirzasoleiman, A. Karbasi, and A. Krause. Deletion-robust submodular maximization: Data summarization with "the right to be forgotten". In *ICML*, 2017.

[35] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions –I. *Math. Program.*, 14(1), 1978.

[36] The news 2013 data set, 2013.

[37] C. Ni, L. Wang, H. Liu, C.-C. Leung, L. Lu, and B. Ma. Submodular data selection with acoustic and phonetic features for automatic speech recognition. In *ICASSP*. IEEE, 2015.

[38] X. Peng, L. Wang, X. Wang, and Y. Qiao. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *Computer Vision and Image Understanding*, 2016.

[39] V. Rocková, G. Moran, and E. George. Determinantal regularization for ensemble variable selection. In *AISTATS*, 2016.

[40] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis*, 115(3):211–252, 2015.

[41] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. Technical report, Cornell University, 1987.

[42] F. Sebastiani. Machine learning in automated text categorization. *CSUR*, 34(1):1–47, 2002.

[43] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *preprint arXiv:1312.6229*, 2013.

[44] R. Sipos, P. Shivaswamy, and T. Joachims. Large-margin learning of submodular summarization models. In *EACL*, pages 224–233. ACL, 2012.

[45] R. Sipos, A. Swaminathan, P. Shivaswamy, and T. Joachims. Temporal corpus summarization using submodular word coverage. In *ICML*. ACM, 2012.

[46] M. Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004.

[47] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: A large margin approach. In *ICML*. ACM, 2005.

[48] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(11):1958–1970, 2008.

[49] G. Trigeorgis, K. Bousmalis, S. Zafeiriou, and B. Schuller. A deep semi-nmf model for learning hidden representations. In *ICML*, 2014.

[50] S. Tschiatschek, R. K. Iyer, H. Wei, and J. A. Bilmes. Learning mixtures of submodular functions for image collection summarization. In *NIPS*, 2014.

[51] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *JMLR*, 6(Sep):1453–1484, 2005.

[52] Z. Wan, H. He, and B. Tang. A generative model for sparse hyperparameter determination. *IEEE Trans. Big Data*, 4(1):2–10, 2018.

[53] K. Wei, R. Iyer, and J. Bilmes. Fast multi-stage submodular maximization. In *ICML*, 2014.

[54] K. Wei, Y. Liu, K. Kirchhoff, C. Bartels, and J. Bilmes. Submodular subset selection for large-scale speech training data. In *ICASSP*. IEEE, 2014.

[55] K. Wei, Y. Liu, K. Kirchhoff, and J. Bilmes. Using document summarization techniques for speech data subset selection. In *HLT-NAACL*, 2013.

[56] K. Wei, Y. Liu, K. Kirchhoff, and J. Bilmes. Unsupervised submodular subset selection for speech data. In *ICASSP*. IEEE, 2014.

[57] J. Xu, L. Mukherjee, Y. Li, J. Warner, J. M. Rehg, and V. Singh. Gaze-enabled egocentric video summarization via constrained submodular maximization. In *CVPR*. IEEE, 2015.

[58] R. R. Yager and J. P. Espada. New advances in the internet of things. 2017.

[59] C.-N. J. Yu and T. Joachims. Learning structural svms with latent variables. In *ICML*. ACM, 2009.

[60] Y. Yue and C. Guestrin. Linear submodular bandits and their application to diversified retrieval. In *NIPS*, 2011.

A Mechanism for Reducing the Number of Mixture Components in F_w

The mixture F_w is composed of several "mother" functions paired with scaling factors Γ. On the one hand, having a rich set of Γ calibration parameters gives the mixture more opportunity to ensure that the information in the modular functions m_u is used appropriately. On the other hand, it increases the burden on the mixture learning in Section 3 since there are more components to learn. Moreover, indiscriminately choosing these components and calibration coefficients can result in components that are highly correlated and hence are themselves redundant. In order to be more efficient, it is beneficial to have a diverse set of components. To this end, we develop a component selection and summarization method such that the resultant components themselves are diverse. We use submodularity for this task but instead of images or documents etc., the objects being summarized are the instantiated mother functions.

Our process is as follows. Let, for example, the set of mother functions and the calibration coefficients factors be: 1) power functions of the form $x^{0.5}$,$x^{0.6}$,$x^{0.7}$,$x^{0.8}$ and $x^{0.9}$ with $\gamma_{pow} \in \{1\}$; 2) log functions of the form $log(1+\gamma_{lg}x)/log(1+\gamma_{lg})$ with $\gamma_{lg} \in \{0.1, 0.2, 0.5, 1, 2, 5, 10, 30, 50, 75, 100, 175, 200, 225, 275, 300\}$; 3) ratio functions of the form $(1 - 1/\gamma_{ra}x)/(1 - 1/\gamma_{ra})$ with $\gamma_{ra} \in \{1.125, 1.5, 2, 5, 10, 30, 50, 75, 100, 175, 200, 225, 275, 300\}$; 4) and saturation functions of the form $min(\gamma_{sat}x, 1)$ with $\gamma_{sat} \in \{1.125, 1.5, 2, 5, 10, 30, 50, 75, 100, 175, 200, 225, 275, 300\}$. This gives 51 mother function-scaling factor combinations. Let this be a ground set $\hat{V}$.

We generate the evaluations of each $\hat{v} \in \hat{V}$ for random subsets of the actual data V (e.g., images in the case of image summarization). We use these evaluations to generate a correlation coefficient matrix Σ. Here $\Sigma_{i,j} = \frac{C_{ij}}{\sqrt{C_{ii}C_{jj}}}$, where C is the covariance matrix. Since Σ can contain negative values, we use instead the absolute values, i.e., a matrix $\hat{\Sigma}$ whose (i,j) element $\hat{\Sigma}_{i,j} = ||\Sigma_{i,j}||$. Next, use the matrix $\hat{\Sigma}_{i,j}$ to instantiate a facility location function of the form $g(\hat{A}) = \sum_{i \in \hat{V}} \max_{j \in \hat{A}} \hat{\Sigma}_{i,j}$ for $\hat{A} \subseteq \hat{V}$. The top L diverse elements in $\hat{V}$ can now be selected by using a greedy procedure for maximizing $g(\cdot)$.

Let I be the index set of all the different concave functions in use (power, log, ratio and saturation functions in the above example) . Then the mixture $F_w(\cdot)$ can be written as:

(A.1)

$$F_w(A) = \sum_{i \in I} \sum_{u \in U} \sum_{\gamma \in \Gamma_i} w_{i,u,\gamma} \frac{\phi_{i,u}\left(\gamma \frac{\sum_{a \in A} \mathfrak{e}(x_a)(u)}{\sum_{v \in V} \mathfrak{e}(x_v)(u)}\right)}{\phi_{i,u}(\gamma)} = \sum_{i \in I} \sum_{u \in U} \sum_{\gamma \in \Gamma_i} w_{i,u,\gamma} \phi_{i,u,\gamma}(m_u(A)).$$

This expanded form of Equation (2.2) explicitly incorporates the different types of concave functions. Furthermore, note that in the experiments, for a given $i \in I$, $\phi_{i,u}(\cdot)$ are the same concave functions $\forall u \in U$.

B Component Mother Functions Used In Experiments

After component summarization as described in Appendix A, we are left with an essential set of components, and this was done separately for each of the three data instances, since in each case we start with a separate modular function. Below we describe the resulting components in each case. Separate summarization was done under the resulting AC features and non AC features in each case.

B.1 Image Summarization The original set of mother functions were: 1) power functions of the form $x^{0.5}$,$x^{0.6}$,$x^{0.7}$,$x^{0.8}$ and $x^{0.9}$; 2) log function of the form $log(1 + \gamma_{lg}x)/log(1 + \gamma_{lg})$; 3) ratio function of the form $(1 - 1/\gamma_{ra}x)/(1 - 1/\gamma_{ra})$; 4) and saturation function of the form $min(\gamma_{sat}x, 1)$. The original set of calibration coefficients are: 1) $\gamma_{pow} \in \{1\}$ for power functions; 2) $\gamma_{lg} \in \{0.1, 0.2, 0.5, 1, 2, 5, 10, 30, 50, 75, 100, 175, 200, 225, 275, 300\}$ for log functions; 3) $\gamma_{ra} \in \{1.125, 1.5, 2, 5, 10, 30, 50, 75, 100, 175, 200, 225, 275, 300\}$ for ratio function; 4) $\gamma_{sat} \in \{1.125, 1.5, 2, 5, 10, 30, 50, 75, 100, 175, 200, 225, 275, 300\}$ for saturation function. The number of random subset samples used to compute the correlation coefficient matrix was 14000.

The component functions discovered for AC features were: 1) a single power function of the form $x^{0.5}$ with $\gamma_{pow} \in \{1\}$ (γ normalization is ineffectual in this case); 2) log functions of the form $log(1 + \gamma_{lg}x)/log(1 + \gamma_{lg})$ with $\gamma_{lg} \in \{0.2, 300\}$; and 3) saturation functions of the form $min(\gamma_{sat}x, 1)$ with $\gamma_{sat} \in \{5, 10, 30, 50, 75, 100, 225\}$.

The component functions for the non AC case were: 1) a single power function of the form $x^{0.5}$ with $\gamma_{pow} \in \{1\}$; 2) log functions of the form $log(1 + \gamma_{lg}x)/log(1 + \gamma_{lg})$ with $\gamma_{lg} \in \{0.2, 300\}$; and 3) saturation functions of the form $min(\gamma_{sat}x, 1)$ with $\gamma_{sat} \in \{5, 10, 30, 50, 75, 100, 200\}$.

B.2 Document Summarization The original set of mother functions were: 1) power functions of the form $x^{0.5}$,$x^{0.6}$,$x^{0.7}$,$x^{0.8}$ and $x^{0.9}$; 2) log function of the form $log(1 + \gamma_{lg}x)/log(1 + \gamma_{lg})$; 3) and saturation function of the form $min(\gamma_{sat}x, 1)$. The original set of calibration coefficients are: 1) $\gamma_{pow} \in \{1\}$ for power functions; 2) $\gamma_{lg} \in \{0.1, 0.2, 0.5, 1, 2, 5, 10, 30, 50, 75, 100, 175, 200, 225, 275, 300\}$ for log functions; 3) $\gamma_{sat} \in \{1.125, 1.5, 2, 5, 10, 30, 50, 75, 100, 175, 200, 225, 275, 300\}$ for saturation function. The number of random subset samples used to compute the correlation coefficient matrix was 10000.

The summarized components for this task for AC case were: 1) log functions of the form $log(1 + \gamma_{lg}x)/log(1 + \gamma_{lg})$ with $\gamma_{lg} \in \{1.5, 5, 300\}$; and 2) saturation functions of the form $min(\gamma_{sat}x, 1)$ with $\gamma_{sat} \in \{2, 5, 10, 30, 50, 100, 225\}$.

The component functions for the non AC case were: 1) log functions of the form $log(1+\gamma_{lg}x)/log(1+\gamma_{lg})$ with $\gamma_{lg} \in \{1.5, 30, 300\}$; and 2) saturation functions of the form $min(\gamma_{sat}x, 1)$ with $\gamma_{sat} \in \{2, 5, 10, 30, 50, 100, 225\}$.

B.3 Subset Selection for ML Training The original set of mother functions were: 1) power functions of the form $x^{0.5}$,$x^{0.6}$,$x^{0.7}$,$x^{0.8}$ and $x^{0.9}$; 2) log function of the form $log(1 + \gamma_{lg}x)/log(1 + \gamma_{lg})$; 3) and saturation function of the form $min(\gamma_{sat}x, 1)$. The original set of calibration coefficients are: 1) $\gamma_{pow} \in \{1\}$ for power functions; 2) $\gamma_{lg} \in \{0.1, 0.2, 0.5, 1, 2, 5, 10, 30, 50, 75, 100, 175, 200, 225, 275, 300\}$ for log functions; 3) $\gamma_{sat} \in \{1.125, 1.5, 2, 5, 10, 30, 50, 75, 100, 175, 200, 225, 275, 300\}$ for saturation function. The number of random subset samples used to compute the correlation coefficient matrix was 1000.

The summarized components for this task in the AC case were: 1) power function of the form $x^{0.8}$; 2) log functions of the form $log(1+\gamma_{lg}x)/log(1+\gamma_{lg})$ with $\gamma_{lg} \in \{300\}$; and 3) saturation functions of the form $min(\gamma_{sat}x, 1)$ with $\gamma_{sat} \in \{30, 75, 175, 275\}$.

The component functions in the non AC case were: 1) power function of the form $x^{0.8}$; 2) log functions of the form $log(1+\gamma_{lg}x)/log(1+\gamma_{lg})$ with $\gamma_{lg} \in \{300\}$; and 3) saturation functions of the form $min(\gamma_{sat}x, 1)$ with $\gamma_{sat} \in \{10, 50, 75, 175\}$.

C Meta-Objective Discrete Optimization

The constrained maximization used in $J_1(w)$, $J_3(w)$, $J_5(w)$, and $J_6(w)$ in Section 3 is an NP-Hard problem. However, a greedy algorithm can provide a $1 - 1/e$ approximation in polynomial time [35]. The most basic version of the algorithm involves iteratively adding elements to a candidate set up until the budget constraint is exceeded. The algorithm starts with an empty set and at each iteration adds the element having largest function gain.

The constrained minimization used in $J_1(w)$ involves the usage of semidifferential based techniques proposed in [17]. For a function $f(\cdot)$, supergradients are defined to formulate a modular approximation of $f(\cdot)$ in the form $m^{g_Y}(X) = f(Y) + g_Y(X) - g_Y(Y) \geq f(X)$. Here g_Y is the generic super gradient of $f(\cdot)$ at Y. This formulation is then used with constraints $\mathscr{C}$ in the following form:

Algorithm 1 MMin procedure by [17]

```
Start with arbitrary $X^0$
**repeat**
    Choose a super gradient $g_{X^t}$ at $X^t$
    $X^{t+1} \in \text{argmin}_{X \in \mathscr{C}} m^{g_{X^t}}(X)$
    t = t + 1
**until** Convergence
```

D Tuning of the Meta-hyperparameters

The objective function $J(w)$ in Section 3.7 (Eq. (3.10)) consists of several meta-objectives $J_i(w)$ combined using the meta-hyperparameters $\{\lambda_i\}_i$. These $\{\lambda_i\}_i$ parameters were tuned using random search [3, 2]. For image summarization the tuning was performed using cross validation. In the case of document summarization the tuning was done using the training set DUC 03. For the MNIST based training set summarization task the parameters were tuned on the created validation set V_{val}. In all cases, hyperparameter tuning did not use any information about the final test set at all.

The meta-hyperparameter (λ) values for the reported results are shown in Tables 1, 2, and 3.

E Additional Analysis of AC Features in the Image Case

The results above demonstrate that AC features can perform better than the non AC (unconstrainedly trained) features for image and document summarization and also for training set summarization task through feature based functions. In this section, we offer further analysis regarding the nature of AC features on image sets for which visualization is relatively easy.

To this end we train a 48 layer autoencoder with a 128 dimensional bottleneck ($|U| = 128$) to generate AC features on the TinyImage data set [48]. The TinyImage data set consists of $\sim$80 million small images. The data set has high redundancy and hence is beneficial for both summarization and feature analysis.

Due to the depth of the network, it is possible that the generated features correspond to abstract concepts. However, we attempt to explore any commonality between images that have high values for the same feature

Table 1: Meta-hyperparameters used in image summarization

Bottleneck	UM(AC)	UM(nonAC)
25	(0.0509, 0.3766, 0.0441, 0.2512, 0.2288, 0.0484)	(0.0437, 0.3101, 0.0532, 0.1524, 0.2983, 0.1424)
98	(0.0942, 0.0126, 0.1248, 0.2294 0.3513, 0.1877)	(0.0122, 0.4338, 0.0376, 0.0425, 0.3684, 0.1054)
242	(0.0281, 0.3068, 0.0648, 0.2816, 0.2298, 0.0889)	(0.0831, 0.2613, 0.0201, 0.3247, 0.2065, 0.1043)

Table 2: Meta-hyperparameters used in document summarization

Bottleneck	UM(AC)	UM(nonAC)
5	(0.1984, 0.3051, 0.0569, 0.0364 0.2941, 0.1092)	(0.2563, 0.4403, 0.0376, 0.0647, 0.0244, 0.1767)
25	(0.5654, 0.1663, 0.0274, 0.1031, 0.0540, 0.0838)	(0.3200, 0.2094, 0.0022, 0.0086, 0.3214, 0.1384)
50	(0.4192, 0.0281, 0.0004, 0.3761, 0.1189, 0.0571)	(0.0163 0.2936, 0.0041, 0.2706, 0.2167, 0.1987)

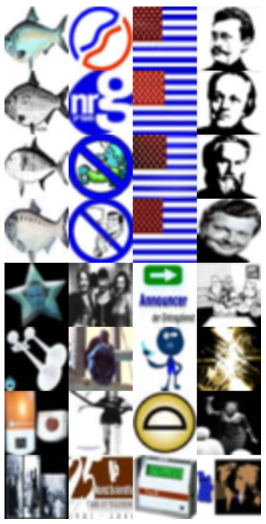


Figure 3: Each column corresponds to an individual feature dimension of the AC features and consists of the top images (based on feature value) for that dimension in the upper half and the bottom images in the lower half from a subset of the TinyImage data set. A subset of the AC feature dimensions ($|U| = 128$) is depicted in this image.

dimension. We generated the AC features for a subset of the TinyImage data set (~4 million images). For each feature dimension, a sorted list of images was produced based on the feature value of the image for the given feature dimension. We collated the top images (high feature value) and bottom images (low feature value) for some of the feature dimensions. The montage of these images is shown in Figure 3. It can be seen that for each dimension the top images (in the upper half of Figure 3) have some similarity. For example, in the first column, all top images have shapes related to fishes. Similarly, the last column has several faces. This demonstrates that the AC features are able to depict informative concepts that can be beneficial for summarization. The bottom four images in each column should indicate none of the properties of the top four. For example, four bottom images of the left column should show no "fish"-like properties, and the four bottom images of the right column should show no "face"-like properties. Visual inspection suggests this is indeed the case.

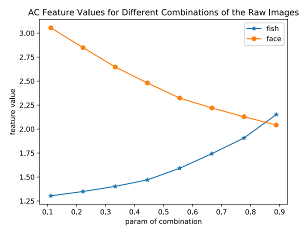
Next we explore if the AC features so produced behave as they should (i.e., a high quantity of a concept in the raw image produces a higher value for the corresponding feature dimension). We explore this by mixing two images (a fish and a face from Figure 3) in a convex combination and observing the corresponding feature values. Figure 4c depicts the resultant image at different stages of mixing. Figure 4a plots the corresponding values of the 'fish' and 'face' feature dimensions (the dimensions corresponding to column 1 and column 4 in Figure 3). It can be observed that value of the 'fish' dimension grows with the increase of fish in the image. Similarly the value of the 'face' dimension diminishes with the decrease in the amount of face in the image. Indeed, this is the desired behavior demonstrating the AC characteristic of the features. To make this more concrete, lets say that the model has a 'fish' feature x_f. In the AC constraint, there is a monotonicity and non-negativity property associated with the feature, in that: a) we have that $x_f >= 0$, and if we have two images, say x, and x', where x' is more 'fish-like', then $x'_f > x_f$ (so x_f increases as the 'fish ness' of x increases). Critically, however, as seen in Figure 4b this is not necessarily the case for non AC features (identical architecture of the neural net, but trained without the AC constraint). This helps to explain why the AC features perform better for summarization objectives in Section 5.

We also explored if empirically the AC feature training caused the constrained matrix W to become identity. An example for the 1936 bottleneck size case is shown in Figure 5. It can be seen that this is not the case.

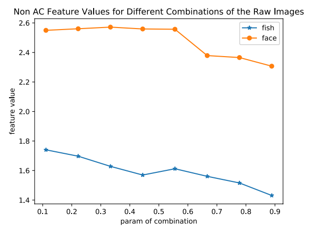
F Sample architectures for feature extraction

Table 3: Meta-hyperparameters used in subset selection for ML training

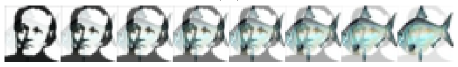
Bottleneck	UM(AC)	UM(nonAC)
4	(0.0266, 0.4144, 0.1329, 0.3119, 0.1089, 0.0053)	(0.0089 0.4301, 0.1300, 0.2874, 0.1228, 0.0209)
16	(0.0310, 0.0011, 0.0783, 0.2746, 0.1488, 0.4662)	(0.4402, 0.0305, 0.0013, 0.4270, 0.0917, 0.0093)
25	(0.2459, 0.0830, 0.0019, 0.2218, 0.1591, 0.2883)	(0.0409, 0.0072, 0.0260, 0.2211, 0.2584, 0.4464)



(a)



(b)



(c)

Figure 4: Plot of the ‘fish’ and ‘face’ feature values for different combination of the raw images. (a) When the features are AC, a higher value of the combination parameter corresponds to a higher contribution from the image of fish and a lower contribution from the image of face in the resultant image. (b) However, in this example, when the features are non AC, a higher feature value does not necessarily correspond to a higher contribution of the corresponding object. (c) The sequence of images corresponding to combinations of the face and fish images.

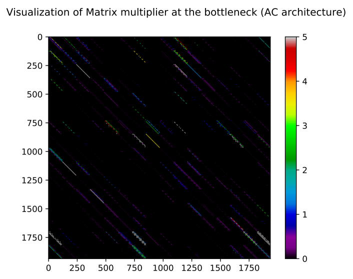


Figure 5: The matrix W following the bottleneck trained with an AC constraint that is neither the identity nor a permutation matrix.

Table 4: Sample Neural network structure (fully connected layers) of autoencoder for extracting features from sentences for the document summarization experiments.

Group	number of nodes
fc1	300
fc2	200
fc3	100
fc4	75
fc5	50
fc6	75
fc7	100
fc8	200
fc9	300
fc10	500

Table 5: Sample Neural network structure (fully connected layers) of autoencoder for extracting features from images for the MNIST data subset selection experiments.

Group	number of nodes
fc1	100
fc2	75
fc3	50
fc4	35
fc5	20
fc6	16
fc7	20
fc8	30
fc9	50
fc10	75
fc11	100
fc12	784

Table 6: Sample Neural network structure of autoencoder for extracting features from images for the image summarization experiments.

Group	Block Type (kernel sz, stride, channels)	# Blocks
conv1	$\left[\ 11 \times 11\ \right]$, 4, 96	1
conv2 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 96	2
conv3	$\left[\ 5 \times 5\ \right]$, 2, 256	1
conv4 (residual)	$\begin{bmatrix} 5 \times 5 \\ 5 \times 5 \end{bmatrix}$, 1, 256	2
conv5	$\left[\ 3 \times 3\ \right]$, 2, 384	1
conv6 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 384	1
conv7	$\left[\ 5 \times 5\ \right]$, 1, 2	1
conv8 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 2	1
deconv9 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 2	1
deconv10	$\left[\ 5 \times 5\ \right]$, 1, 384	1
deconv11 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 384	1
deconv12	$\left[\ 5 \times 5\ \right]$, 2, 256	1
deconv13 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 256	2
deconv14	$\left[\ 4 \times 4\ \right]$, 2, 96	1
deconv15 (residual)	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, 1, 96	2
deconv16	$\left[\ 12 \times 12\ \right]$, 4, 3	1