Learning Personal Style from Few Examples

David Chuan-En Lin
HCI Institute
Carnegie Mellon University
Pittsburgh, PA, USA
chuanenl@cs.cmu.edu

Nikolas Martelaro
HCI Institute
Carnegie Mellon University
Pittsburgh, PA, USA
nikmart@cmu.edu

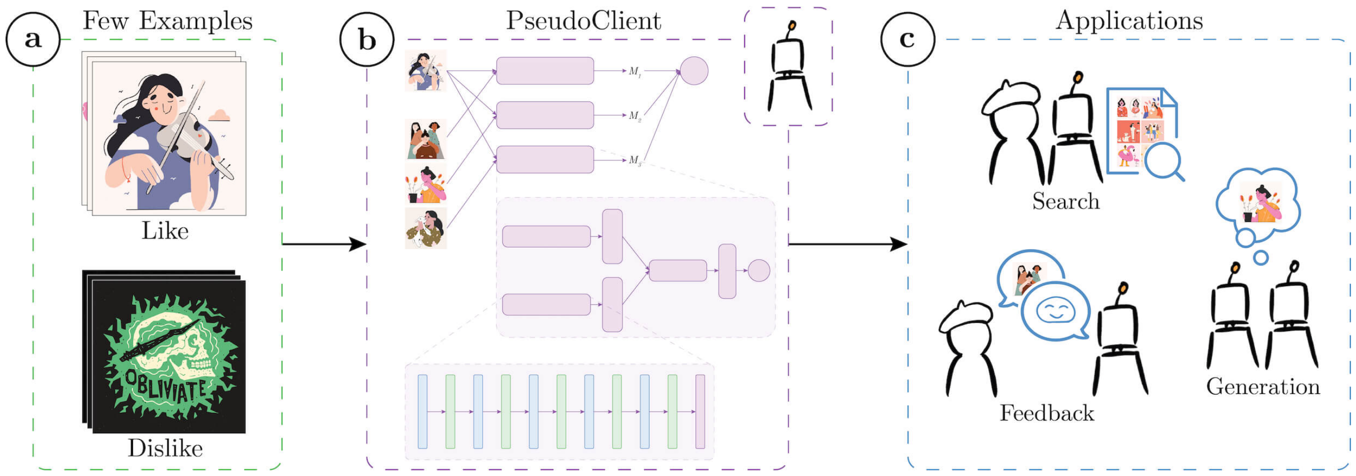

Figure 1: Given a few design examples (a), PseudoClient learns a computational model of the client's personal style preferences (b) to support multiple practical design applications (c).

ABSTRACT

A key task in design work is grasping the client's implicit tastes. Designers often do this based on a set of examples from the client. However, recognizing a common pattern among many intertwining variables such as color, texture, and layout and synthesizing them into a composite preference can be challenging. In this paper, we leverage the pattern recognition capability of computational models to aid in this task. We offer a set of principles for computationally learning personal style. The principles are manifested in PseudoClient, a deep learning framework that learns a computational model for personal graphic design style from only a handful of examples. In several experiments, we found that PseudoClient achieves a 79.40% accuracy with only five positive and negative examples, outperforming several alternative methods. Finally, we discuss how PseudoClient can be utilized as a building block to support the development of future design applications.

CCS CONCEPTS

• **Human-centered computing → Human computer interaction (HCI)**; • **Applied computing → Arts and humanities**.

KEYWORDS

style, personal preference, graphic design, machine learning

ACM Reference Format:
David Chuan-En Lin and Nikolas Martelaro. 2021. Learning Personal Style from Few Examples. In *Designing Interactive Systems Conference 2021 (DIS '21), June 28-July 2, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3461778.3462115

1 INTRODUCTION

A key role of the designer is being able to develop a firm grasp of the client's implicit tastes [4]. For example, in the case of graphic design, a hotel group may favor a minimal and luxurious feel, while an outdoor apparel company may prefer a more bold and rugged touch. However, since clients are rarely trained to constructively articulate their design ideas, their comments may often be vague and difficult to implement [56].

"Make it pop... is the dumbest thing you could say to a graphic designer."
— Reddit user HylianHandy

To gain an understanding of the client's tastes, many designers arrange early meetings with clients to probe into their personal style preferences. Some designers use specialized tools to help with this process of extracting implicit information from the clients, such as Brand Deck [2]. Brand Deck consists of 100 adjective cards (e.g., vibrant, futuristic, historic) that the designer asks the client to go through and sort into three piles: you are, you are not, and does not apply. One caveat of using a tool like Brand Deck is that designers will need to further examine whether the client's understanding of subjective words matches with theirs. Other designers prefer to give clients more visual examples, such as a logo book [14] or a mood board [34] consisting of a collage of different visual samples, and asking the client to pick a few samples that resonate with them. By using a selection of images, clients won't need to articulate their preferences through subjective words, but rather, by simply picking what they visually prefer. However, even when the client provides a pool of examples, recognizing a common pattern among numerous variables such as color, texture, and layout, synthesizing them into a composite understanding of the client's style preference, and using this understanding to generate a new design is a non-trivial task [15]. In this paper, we ask: can computational systems be used to aid in the task of learning peoples' style preferences from a set of example images?

Motivated by findings that high-level perceptions of a design are correlated with low-level features of appearance attributes [60], recent works seek to identify such low-level features using a computational approach to assist designers. Several works have explored extracting the underlying structure of designs such as the Document Object Model (DOM) tree of a web page [9, 31]. However, such methods only work with a small subset of design categories, such as webpages or vector images, where there is an encoded and explicit structure. For graphic design, where only bitmap information is available instead of structural elements such as shapes and text, other research has explored engineering various hand-crafted image features such as color histograms and edge maps [26] for predicting visual style. However, akin to the limitations of rule-based AI, hand-crafted features are limited in representation power and generalizability as some appearance attributes may be too complex or abstract to manually define. More recent works have experimented with using neural networks to extract a model of visual style due to their strong ability in automatically and adaptively learning the most relevant low-level features [24, 59]. In our work, we look to use neural networks to automatically learn relevant features, but with two key distinctions. First, unlike most data-hungry machine-learning-based methods, our method only needs a small handful of examples. Second, prior work mostly attempts to categorize content into subjective style terms (e.g., futuristic, minimalist) [8, 59] based on the preferences of the crowd. Conversely, our work focuses on learning an individual's *personal* style preference, without ever having to put a word on it nor relying on generalized conceptions of design style.

We introduce a set of principles for learning a client's personal style from few examples. We then use these principles to develop PseudoClient, a deep learning framework that takes in a handful of examples and automatically learns a computational model of the client's personal style. Our use case focuses on graphic design, where we feed graphic design samples as input. We first ask the client to select a few examples they like and dislike. Using the samples provided, PseudoClient learns the client's personal style preferences using a metric learning approach [30] with twin Convolutional Neural Networks [29]. Exploiting the benefits of metric learning with deep neural networks, PseudoClient requires only a few examples, does not rely on limited hand-crafted features, and can be retrained quickly with additional samples. We conduct experiments to assess how PseudoClient compares to other methods as well as how it may be affected by various factors such as training sample size and the ratio of positive and negative samples. We find that PseudoClient outperforms our baseline methods and can be tailored for different needs. Finally, we discuss PseudoClient's capability of supporting the development of future design tools in three directions: search, feedback, and generation.

In summary, our contributions are three-fold:

- A set of principles for learning personal style from few examples. The principles are manifested in PseudoClient, a novel deep learning framework for learning an individual's graphic design style from a handful of examples and without relying on generalized conceptions of subjective style.
- Quantitative and qualitative experimental results demonstrating PseudoClient's advantages compared to other methods. We further examine how number of examples and ratio of positive and negative examples affect performance.
- A discussion of potential applications where PseudoClient can support augmenting designers' capabilities. We illustrate how PseudoClient is useful as a building block for multiple practical design applications by exploring three areas of design work.

2 RELATED WORK

Our work is situated among literature in computational design understanding. More specifically, our research builds on prior work in two main branches: assessing aesthetics as a quality indicator and more fine-grained understanding of artistic style [46].

2.1 Assessing Aesthetic Quality

Computational methods of assessing the aesthetic quality of designs have a wide range of applications spanning from media editing to curation. Such methods typically involve defining a set of descriptors for some content and using them to predict human-labeled aesthetic ratings. Michailidou et al. [36] analyzes statistics of web page elements such as number of images, words, and links and Reinecke et al. [40] further incorporates page-level statistics such as number of leaves in a quadtree decomposition. In applications where the underlying structural information of the design is not available, such as bitmap photographs, prior works have engineered various visual features. Datta et al. [10] extracts 56 features based on photography concepts such as rule-of-thirds and depth-of-field. Isola et al. [25] investigates the correlation between image memorability and a set of high-level visual features such as object and scene semantics.

However, handcrafted features are nevertheless limited as some important aspects of a design may be too elusive or complex to manually define. Therefore, work in recent years has investigated the use of neural networks to automatically learn the most relevant low-level features for pattern finding. For example, NIMA [50], one of the top performers on the Aesthetic Visual Analysis dataset [37], explores using CNNs to automatically discover important features without human supervision. In our work, we also exploit the strength of neural networks in feature extraction and do not use manual feature engineering. Rather than predicting an aesthetics rating, we predict a match score: how well does a design align with an individual's *personal* preferences as opposed to the weighted average of preferences from the crowd.

2.2 Understanding Style

Closely related to evaluating aesthetic quality is the task of understanding artistic style. We loosely categorize prior work in this domain into two approaches: tagging, where the task is to automatically assign labels to some content, and similarity, where the task is to learn the similarity across different content. Note that the two approaches are not necessarily mutually-exclusive.

Tagging has been extensively studied in information retrieval and recommender systems communities [11] although fewer works have explicitly approached tagging in terms of artistic *style*. Prior work in tagging by style has largely treated it as some form of classification problem. Shamir et al. [45] assigns paintings to painters and schools of art using image descriptors such as color histograms and edge statistics. Karayev et al. [26] predicts style labels such as bright and energetic for photographs and paintings using features such as color histogram and visual saliency. Wu et al. [55] models the brand personalities of mobile UIs with UI descriptors such as color, organization, and texture. Vaccaro et al. [51] predicts high-level fashion styles attributes such as tropical and exotic from low-level design element language such as color and material using polylingual topic modeling. More recent works have explored style tagging with deep neural networks. Zhao et al. [59] characterizes personalities for graphic designs such as cute and mysterious. Takagi et al. [49] proposes an expert-curated fashion style dataset containing 14 categories such as rock and street and found that modern computer vision classification networks are able to outperform fashion-naïve users but not fashion-savvy users. However, since style tags are intrinsically subjective, labeling content with style tags is by nature a noisy task. Thus, our work does not aim to describe personal style with subjective style terms. Rather, our tags are simply whether a design "fits" or "does not fit" with an individual's personal style.

The question of "does a design fit or not fit with an individual's personal style?" can also be reformulated as "how similar is a design when compared to the individual's personal style?" This formulation opens up an interesting repertoire of research in "similarity by style" to draw inspiration from. D.Tour [41] and Webzeitgeist [31] allow search by stylistic similarity based on features of a web page such as DOM tree depth and number of leaf nodes. Several other works on learning similarity by style include applications in infographics [42], illustrations [18], icons [32], fonts [38], fashion [47], and 3D furniture models [33]. In our work, we borrow from the concept of similarity by style to define personal style based on similarity with a set of representative examples selected by the individual. Prior works have explored various methods for computing similarity, such as through color histograms [22] and Convolutional Neural Networks [54] (also see Comparison with Baselines subsection). Recent works in the Machine Learning community have shown the effectiveness of metric learning approaches [30, 35] for areas such as fashion [52] and home goods product design [6]. We build upon the successes of metric learning to design a metric learning framework for personal graphic design style. To guide our work, we distill a set of principles.

3 PRINCIPLES OF LEARNING PERSONAL STYLE

To support our objective of learning personal style preferences, we ground the development of PseudoClient in four central principles.

3.1 Principle 1: Learn by Example

Considering that design vocabulary (e.g., minimal, vintage) is inherently subjective and highly dependent and interpreted based on the individual's knowledge and experiences [46], we do not ask clients to indicate their preferences through such vocabulary nor do we attempt to fit their preferences into such vocabulary. Inspired by the mood board technique [34], we simply ask the client to supply us with visual examples and learn to judge the client's personal style based on them.

3.2 Principle 2: Learn by a Handful

Since our task is to learn personal style, our examples come directly from the client of interest. However, asking the client to select massive amounts of examples is tedious. In addition, prior works have found that as the quantity of examples required from the client increases, the quality and consistency of the examples begin to decrease due to fatigue and the pressure to reach the target of providing a high number of samples [27]. Based on these observations, rather than being data-hungry, PseudoClient should be capable of working with only a handful of examples.

3.3 Principle 3: Learn by Juxtaposition

Findings from prior work in recommender systems [5] suggest that it is easier to select positive and negative samples than to discern likeability on a spectrum. Given this, we ask the client to provide us a set of examples they like (positive samples) and another set of examples they dislike (negative samples). Our task then is to determine whether a design fits the positive samples or the negative samples more closely. With only a limited number of examples to learn from, learning by juxtaposition allows us to formulate our problem of modeling the client's style preferences into more simplistic binary classification problems.

3.4 Principle 4: Learn by Multiple Comparisons

Studies in the learning sciences and cognitive psychology have observed that through comparison against multiple examples, one can quickly learn a common underlying structure, even if the individual examples are not fully understood [19, 20]. Building on this insight, we repeatedly do pairwise comparisons between the unseen design and the pool of reference examples provided by the client. For a more detailed description and diagram of the method, please refer to the Comparison Framework subsection.

4 IMPLEMENTATION

Our four principles are manifested in PseudoClient and guide its implementation. To the best of our knowledge, we uniquely frame our task of modeling the client's *personal graphic design style* as a metric learning problem [30]. Given an unseen graphic design, our objective is to determine its *similarity* with a set of representative examples selected by the client. This allows us to then classify the

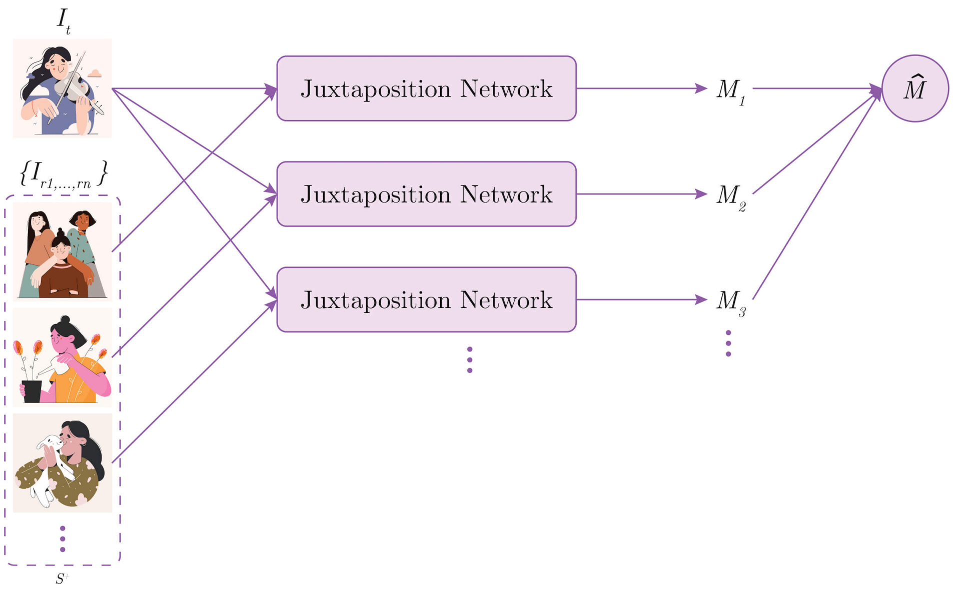

Figure 2: The Comparison Framework. We compute pairwise match scores between the test image and each reference image in the positive support set. The final match score is the median.

design between two classes: client likes and client dislikes. The following outlines the implementation of PseudoClient, including (1) the Support Set, (2) the Comparison Framework, (3) the Juxtaposition Network, (4) the Embedding Network, and (5) the training setup. We wrote our learning framework in PyTorch version 1.6.0.

4.1 Support Set

We take inspiration from the mood board technique [34] by first asking the client to provide a small selection (Principle 2) of graphic design examples (Principle 1). More specifically, we ask the client to pick a few samples they like (positive samples) and another few samples they dislike (negative samples) (Principle 3). This is our support set S. We denote a subset of S containing only positive samples as the positive support set S^+ and a subset of S containing only negative samples as the negative support set S^-.

4.2 Comparison Framework

Figure 2 visualizes the Comparison Framework. Given an unseen graphic design I_t and a positive support set S^+, we perform pairwise comparisons (Principle 4) between I_t and each reference sample $I_{r1,...,rn} \in S^+$ to compute their respective match scores $M_{1,...,n}$ through our Juxtaposition Network (see Juxtaposition Network subsection on how M is computed). A high match score means that I_t is predicted to be in the same class as I_r. This implies that the unseen design I_t matches the client's personal style, since I_r is a sample that the client likes. Conversely, a low match score means that I_t is predicted to be in a different class as I_r. This implies that the unseen design I_t does not match the client's personal style. We are therefore learning to classify designs *indirectly* by evaluating their similarities with a set of labeled referencing examples (positive support set S^+). We compute match scores with the positive support set S^+ as opposed to the full support set S based on our findings from empirical pilot testing. We found that the client's negative examples tend to be less consistent, often consisting of many diverging styles the client dislikes. We then take the median of the pairwise match scores $M_{1,...,n}$ as the overall predicted match score $\hat{M}$. We take the median rather than the mean to decrease the effects of outliers.

$$\hat{M}(I_t) = \text{Median}_{I_r \in S^+}(M(I_t, I_r))$$

4.3 Juxtaposition Network

Since we want to work with a small support set (Principle 2), we approach the challenge of learning a model to accurately predict the match score M as a few-shot learning task [17]. Prior work by Melekhov et al. [35] in image matching showed the strength of Siamese Networks [29] in generalizing from small datasets. We build upon this and learn the similarity function between two graphic design inputs. We design a Juxtaposition Network resembling a Siamese Network with twin Convolutional Neural Networks (CNN).

Figure 3 visualizes the architecture of the Juxtaposition Network. The Juxtaposition Network takes in a test image I_t and a reference image $I_r \in S^+$ as inputs and computes a match score M with *range* = $[0, 1]$ as output. Our Juxtaposition Network uses a twin architecture. We first encode each 3x224x224 RGB input image I_t and I_r into 4096-dimensional feature embeddings $f(I_t)$ and $f(I_r)$ through twin Embedding Networks (see Embedding Network subsection). We then compute the weighted L1-distance D between the two feature embeddings.

$$D(I_t, I_r) = |f(I_t) - f(I_r)|$$

We then translate the distance D into a match score M, which is the probability that the two inputs belong to the same class, by passing it through a fully-connected layer (FC) with learned weights $\mathcal{W}$

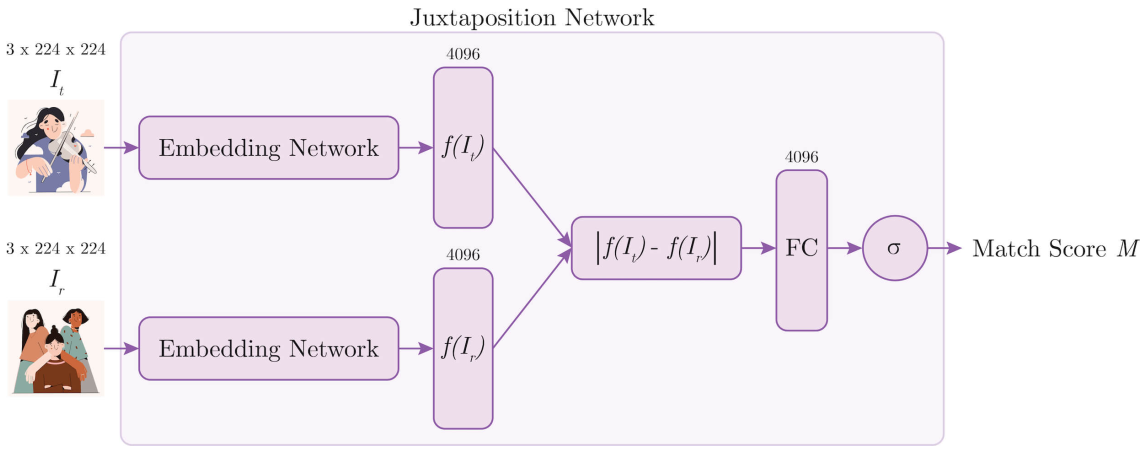

Figure 3: Architecture of the Juxtaposition Network

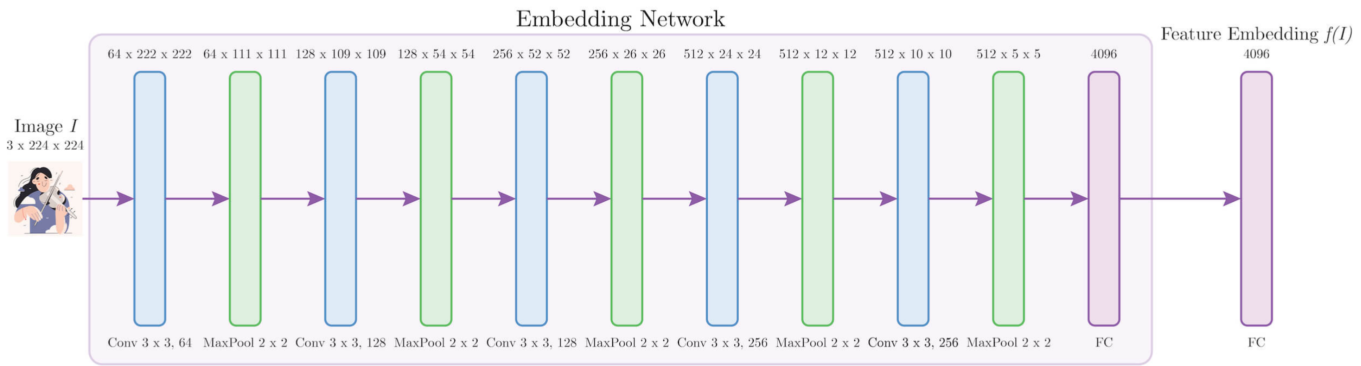

Figure 4: Architecture of the Embedding Network

and a sigmoid activation function (σ).

$$M(I_t, I_r) = \frac{1}{1 + \exp^{-\mathcal{W} \cdot D(I_t, I_r)}}$$

4.4 Embedding Network

Figure 4 visualizes the architecture of the Embedding Network. The Embedding Network takes in a 3x224x224 RGB image as input and extracts a 4096-dimensional feature embedding as output. The feature embedding is a vector representation that captures visual features of the image. We use five convolutional blocks (a 3x3 convolutional layer and a 2x2 max-pooling layer) and two fully-connected layers. We apply batch normalization and ReLU activation after the convolutional layers and sigmoid activation after the fully-connected layers. Our architectural design resembles the well-researched VGG architecture [48] and performs well empirically. While there are certainly techniques to further optimize performance [16], they are not the focus of this paper.

4.5 Training Setup

Our training setup consists of two stages: pre-training and finetuning.

4.5.1 Pre-training. To allow our model to quickly learn a client's personal style with few examples, we first pre-train a network on a dataset of graphic design images collected from dribbble.com. The dataset consists of six different classes (flat, geometric, line, minimal, pop, vintage) with 50 images each, yielding a collection of 300 images. We collected the dataset based on querying relevant keywords, with class labels self-tagged by the artists. Note that the main purpose of pre-training is to seed the network, letting our model first learn some basic visual features of graphic design, such as edges, patterns, or general shapes, an interesting property of neural networks [58]. Thus, our focus is *not* to bin all graphic design work into these six subjective categories. Rather, when the client trains PseudoClient to model their own tastes, instead of training a model from scratch with completely random initialized weights, the client's model can have a head start by building on the pre-trained model via transfer learning [39], resulting in a much shorter training time.

We process the dataset into *pairs* of graphic design images with an assigned binary label y. If the images in a pair belong to the same class, we set $y = 1$. If the images in a pair belong to distinct classes, we set $y = 0$. The pairs are randomly sampled. We resize the images into 3x224x224 pixels with RGB channels and normalize each color channel with $mean = [0.485, 0.456, 0.406]$ and $std =$

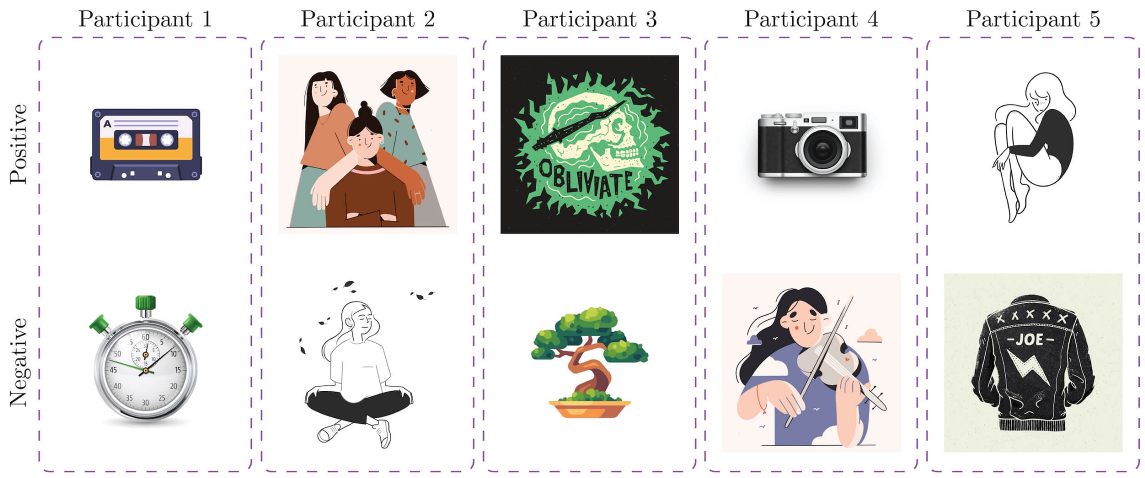

Figure 5: A snapshot of each participants' positive and negative sets. Each column represents a participant. The top row displays an example of their positive set and the bottom row displays an example of their negative set.

[0.229, 0.224, 0.225] based on the statistical distribution of ImageNet [12]. We split our training and validation sets with a 9:1 ratio. We use the Adam optimizer [28], a batch size of 32, a learning rate of 1×10^{-5}, and train for 50 epochs. Since the labels are binary, we use a cross-entropy loss $\mathcal{L}$ for every training batch B.

$$\mathcal{L}(B) = \sum_{I_t, I_r, y \in B} y \log(M(I_t, I_r)) + (1 - y) \log(1 - M(I_t, I_r))$$

Training takes around an hour to complete on an NVIDIA GeForce RTX 2080 Ti graphics card with 11GB of memory.

4.5.2 Fine-tuning. This is the stage where the model learns the client's *personal* preferences. Given a pre-trained model, we fine-tune the model so that it reflects their own tastes. We begin with the pre-trained weights and train the model with respect to their support set S as the new training dataset. Data processing procedures are the same as for the pre-trained model. Therefore, we learn by juxtaposition (Principle 3), where $y = 1$ means that the graphic designs in the pair are both liked or disliked by the client (same class) and $y = 0$ means that one of the graphic designs in the pair is liked by the client while the other is disliked by the client (different class). This means we are not directly learning whether a design is liked or disliked by the client, but rather learning its similarity with both the liked and disliked example sets. We train our model until it can consistently predict the correct y label for each pair. We use the same optimizer and loss function as the pre-trained model, a batch size of 16, a learning rate of 1×10^{-8}, and train for 20 epochs. Training takes around ten minutes to complete on the same hardware as for the pre-trained model.

5 EXPERIMENTS

To evaluate the effectiveness of PseudoClient, we performed several experiments. This serves as a litmus test of the system's performance. The following outlines our setup and various experiments, including:

- **Comparisons with baselines.** We evaluate PseudoClient's performance by comparing against other methods, namely, softmax-based approaches (Convolutional Neural Networks) and traditional distance measurements (color histogram distance).
- **Exploration of various factors.** To discover usage guidelines for future designers and offer an understanding of how designers can work with PseudoClient for their needs, we explore how various factors can affect PseudoClient's ability to learn personal style. We focused on two factors: number of examples (dataset size) and different ratios of positive and negative examples (class imbalance).
- **Query results.** Finally, as graphic design is a highly visual medium, we want to see how PseudoClient performs qualitatively to uncover insights that may not be portrayed through numbers. We do this through a simple image retrieval task, visualizing the retrieval results.

5.1 Setup

Given that our task is to learn personal style preferences, we evaluate our accuracy in doing so with five people: two of the paper's authors and three volunteers. For each participant, we ask them to first come up with a simple design idea (e.g., an illustration for a local French bakery). We then ask them to select 70 images that fit their design idea (positive set) from dribbble.com. This is for our experimental purposes, such that we have enough images for various levels of training and testing. In an actual usage scenario, participants would not need to supply as many examples (see Number of Examples subsubsection).

After all participants finish selecting their positive sets, participants then select another person's positive set (from the pool of positive sets of all other participants) as their negative set. Our requirement is that their selected negative set doesn't fit their design idea. One potential constraint of this approach is that a participant may struggle in selecting a negative set from the pool if all other positive sets match closely with theirs. However, such an issue did not occur during our specific study (maybe because participants had different design ideas). Thus, each positive set was stylistically

Table 1: The accuracies of different methods between participants P1 – P5 and overall. The highest accuracy is highlighted in bold.

Method	P1 Accuracy	P2 Accuracy	P3 Accuracy	P4 Accuracy	P5 Accuracy	Overall Accuracy
Chance	*50.00%*	*50.00%*	*50.00%*	*50.00%*	*50.00%*	*50.00%*
Color Histogram	69.00%	26.00%	53.00%	55.00%	58.00%	52.20%
CNN	67.00%	63.00%	76.00%	53.00%	62.00%	64.20%
PseudoClient (Ours)	**86.00%**	**77.00%**	**90.00%**	**64.00%**	**80.00%**	**79.40%**

different from the negative set. For a snapshot of what the participants picked as their positive and negative sets, please refer to Figure 5. For each positive and negative set of 70 images, we randomly allocate 50 as our *test* set and 20 as our *training* set. This means that 50 of the 70 images will be used as a true test set and not be used for training. For each training set of 20 examples, we further randomly sample various training sets of 1, 5, 10, and 20 examples for our later study on how different numbers of training samples affect performance (see Exploration of Various Factors subsection). For our comparison with baselines and qualitative query results studies, we use the training set of 5 examples (i.e., training with only 5 positive and negative examples). This follows our objective of learning by a *handful* of examples (Principle 2).

We train a separate model for each participant. To measure the accuracy of the model, we then compute a match score for each of the 50 examples in their unseen positive test set as well as for each of the 50 examples in their unseen negative test set, by pairwise comparisons with all examples in their positive training set (see Comparison Framework subsection). Note that the match score M is a numeric value with $range = [0, 1]$ where a higher value implies higher predicted similarity with the positive support set, and a lower value implies a lower predicted similarity with the positive support set. For the positive test set, if M is greater than the threshold of 0.5, we note down a correct true positive (TP) prediction. For the negative test set, if M is less than the threshold of 0.5, we note down a correct true negative (TN) prediction. Our accuracy is then given by $\frac{number\ of\ TP + number\ of\ TN}{100}$. Finally, we take the average of the accuracies for each participant as the overall accuracy.

5.2 Comparison with Baselines

We implemented two baseline models to compare against our method: (1) color histogram and (2) a convolutional neural network (CNN). Our first baseline is based on the distance of color histograms, which is a widely used metric for evaluating similarity between images [22]. We first extract a 3D histogram from each RGB image, using 8 bins per channel and normalize with $range = [0, 256]$. We then flatten the histogram, yielding a 512-dimensional vector. To compute the distance between a pair of vectors, we compute its correlation [3]. The correlation is a numeric value with $range = [0, 1]$, where higher correlation implies smaller distance and more similarity and and lower correlation implies larger distance and less similarity. We compute the overall accuracy of the color histogram method using a similar procedure as the second paragraph of the Setup subsection. However, instead of using a fixed threshold of 0.5, we set the threshold as the mean correlation value between images in the training set since the distribution of correlation values differ greatly across different participants.

Our second baseline is a standard implementation of a CNN, representing a typical neural-network-based approach for classification. We use the architecture of our Embedding Network with a single output node. Rather than computing a match score, we directly classify whether the unseen test image is liked or disliked by the participant. The accuracy is then given by $\frac{number\ of\ correct\ predictions}{100}$ and we also take the average of the accuracies for each participant as the overall accuracy.

By learning to classify designs *indirectly* via learning their similarities with a positive support set, our hypothesis is that PseudoClient would perform better than a standard CNN implementation, given the nature of our task: "whether a design is more similar to the set of like examples or the set of dislike examples" seems more learnable than naively judging "whether a design is liked or disliked." The latter is an intrinsically ambiguous task, since like/dislike more often falls on a scale as opposed to being a perfect dichotomy. In addition, we also hypothesize that color alone would not be sufficient for determining personal style. From Table 1, we observe that color histogram performs only marginally better than random chance. The CNN, which is essentially PseudoClient's Embedding Network, still struggles to classify consistently. Overall, we observe that PseudoClient is able to outperform our two baselines for all participants, supporting our hypotheses.

5.3 Exploration of Various Factors

5.3.1 Number of Examples. We first investigate how supplying different training sample sizes affects PseudoClient's performance. Our hypothesis is that accuracy will increase as the number of examples increases, and we test our hypothesis by training separate models using 5, 10, and 20 positive and negative examples and evaluating their accuracies. We do not explore beyond 20 examples since it goes beyond the definition of "few examples" based on feedback from our participants. We also evaluate how well our pretrained model can generalize to personal style *without* any further fine-tuning and given only 1 reference example.

Table 2 summarizes the accuracies of PseudoClient when given different numbers of examples for each participant and overall. We observe that accuracy generally increases as the number of examples increases, confirming our hypothesis. However, interestingly, when compared to the overall accuracy of the models using 5 examples, the overall accuracy of the models using 10 examples did not increase and even dipped slightly. This may suggest that an increase in performance may only be prompted by a sufficiently large increase in sample size. Another interesting observation is

Table 2: The accuracies of PseudoClient when given different numbers of examples between participants P1 – P5 and overall. The highest accuracy is highlighted in bold.

# Examples	P1 Accuracy	P2 Accuracy	P3 Accuracy	P4 Accuracy	P5 Accuracy	Overall Accuracy
1	47.00%	74.00%	47.00%	53.00%	70.00%	58.20%
5	86.00%	77.00%	90.00%	64.00%	80.00%	79.40%
10	83.00%	73.00%	93.00%	66.00%	75.00%	78.00%
20	**88.00%**	**80.00%**	**96.00%**	**77.00%**	**92.00%**	**86.60%**

Table 3: The overall true positive percentages, true negative percentages, accuracies, and F1 scores of PseudoClient when given different ratios of positive and negative examples. The highest results are highlighted in bold.

Number of Positive:Negative Examples	True Positives	True Negatives	Accuracy	F1 Score
10:5	**82.40%**	68.40%	**75.40%**	**0.77**
5:10	65.20%	**82.80%**	74.00%	0.71

that our pre-trained model, given only 1 guiding example, is able to perform better than random chance by some amount for P2 and P5. The reason may be that the examples chosen by these participants are more uniform, granting the single example a greater representative capacity. Hence, designers should note that clients who have tighter style preferences may not need to supply as many examples.

5.3.2 Ratio of Positive and Negative Examples. Previously, we trained our models with equal amounts of positive and negative examples. We thought it would be interesting to also investigate how performance would be affected if the client gives more positive examples and fewer negative examples, and vice versa. We hypothesize a higher TP when given a higher ratio of positive examples and a higher TN when given a higher ratio of negative examples. We experimented with two setups: 10 positive examples with 5 negative examples and 5 positive examples with 10 negative examples.

Table 3 summarizes the overall true positive percentages, true negative percentages, accuracies, and F1 scores of PseudoClient when given different ratios of positive and negative examples. We observe that the model predicts true positives more consistently when given more positive examples and predicts true negatives more consistently when given more negative examples, supporting our hypothesis. This reveals an interesting property: designers may adjust the ratio of positive and negative examples required for their specific goals. For example, if the goal is targeted towards identifying and filtering out what the client *dislikes*, then the designer may ask the client to focus on supplying more negative examples.

5.4 Query Results

We qualitatively evaluate the performance of PseudoClient with an image retrieval task. We query for the top 10 images with the highest match scores from a database of graphic design samples. Our database consists of a subset of the Dribbble dataset from [8] and some examples selected by the participants. None of the samples were used for training.

Figure 6 visualizes two example queries. The samples bounded by the dashed lines are the positive examples used for training (positive support set) and the samples bounded by the solid lines are the top-10 queried results ranked from 1 to 10. We observe that PseudoClient is able to retrieve stylistically similar graphic designs by synthesizing from a few examples. Interestingly, a couple retrieved designs in Figure 6a even belong to the same artist as the provided examples, demonstrating PseudoClient's capability of recognizing personal design style. Note that the retrieved samples don't necessarily have similar color as the provided examples. For example, the top-ranking retrieval in Figure 6b has a light cream background despite most of the examples having darker backgrounds. Nonetheless, its resemblance to the examples in terms of artistic style is apparent. One limitation we discovered from the queried results is that PseudoClient judges samples in a more holistic sense and may overlook fine-grained but important details such as font styling (see Figure 6b). For example, serif and san-serif fonts may elicit different emotions [44]. We suggest an approach to address this in the Limitations and Future Work section.

6 APPLICATIONS

We suggest possible applications that can be enabled using PseudoClient by illustrating its use cases in augmenting designers from three directions: (1) search, (2) feedback, and (3) generation.

6.1 Search

A natural application of PseudoClient is an example-based, personalized style search engine (Figure 7a). Prior work has shown that being able to find high quality examples is a crucial part of the creative design process for not only gaining inspiration, but also exploring alternatives and performing comparative evaluations [23]. An example usage scenario may be that shown Figure 6. The designer may first request a few examples from the client. Using these examples, the designer may then search for even more examples of similar style. Note that unlike existing search tools built into many design sharing websites, a search engine built on top of PseudoClient has the benefit of searching directly with examples instead of tagging based search with subjective keywords [8]. Furthermore, style-based search can surface designs that have not been tagged with keywords. This setup provides a powerful mechanism

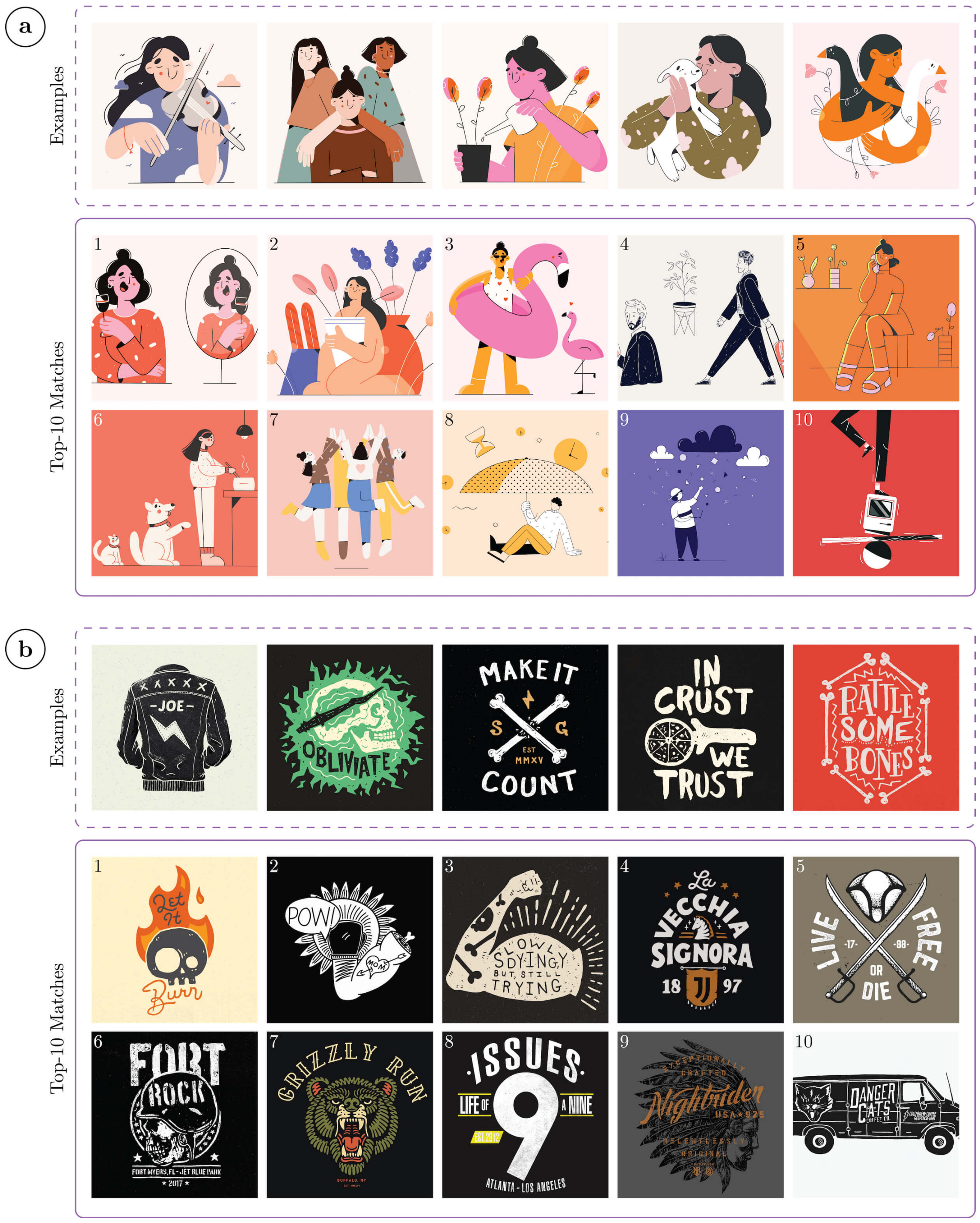

Figure 6: (a) and (b) show two example query results. The dashed lines contain the positive examples used for training (positive support set). The solid lines contain the queried samples with the highest match scores, ranked from 1 to 10.

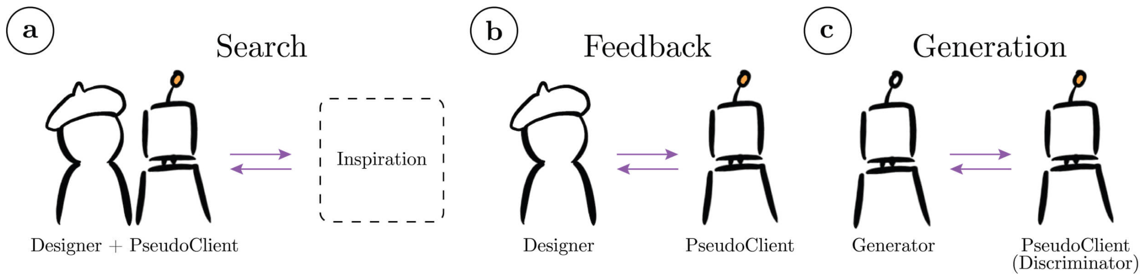

Figure 7: PseudoClient can be used as a building block for multiple practical design applications. We explore applications from three directions: (a) search, (b) feedback, and (c) generation.

for locating potential relevant design examples to draw inspiration from, without having to overwhelm the client with the task of providing large quantities of examples themselves or going through lengthy meetings for designers to probe and synthesize their tastes.

Extending beyond searching for design examples, PseudoClient may also be used for clients to search for *designers*. Since designers often also have their own personal style, it is sensible for clients to find designers who have personal styles that match their tastes. Given a designer's design portfolio, PseudoClient can assess the portfolio's overall similarity with the client's personal style preferences. The client may thus do a "reverse designer search" with PseudoClient to discover the top designers who most fit their needs. Similarly, one may also do a "reverse design community search" with PseudoClient to discover the top design communities that are most well aligned with their personal style preferences. A fellow designer pointed out that being able to cluster designers or design communities by style may also reveal how designers are influenced by one another and unravel interesting design trends and patterns.

6.2 Feedback

The ability of PseudoClient to assess similarity by style opens the possibility of an automatic design feedback system. By referencing the few examples given by the client, PseudoClient can provide rapid, automatic design feedback to the designer for evaluating how well the designer's design drafts align with the client's personal tastes (Figure 7b). Such an application can potentially increase the quality of design work by allowing the designer to receive timely critique for iterative exploration of alternatives and ensure that their design direction remains aligned with the client's tastes, without the constraint of the client's limited availability [13]. This is especially timely as an increasing amount of design work shifts to nowhere and everywhere (remote) where latency between design cycles is substantially magnified due to time zone differences and the absence of face-to-face meetings.

Another interesting use case of PseudoClient's capability of giving feedback, suggested by another fellow designer, is a tutorial system for beginners to mimic the styles of masters.

"Art is sourced. Apprentices graze in the field of culture."

— Jonathan Lethem, Writer

A common way for beginner artists to improve their techniques is by performing master studies. During this process, the beginner studies the techniques of a master by recreating a piece of work using a similar style [1]. However, whether the master copy truly aligns with the master's style may ultimately be difficult to determine. Given a handful of the master's work as examples, PseudoClient can be applied as a simple tutorial system by rating the beginner's master copy. Implementing an activation map to reveal which regions of the copy matches with the master's style may further increase explainability and actionability of PseudoClient's feedback [7]. Such an application can also extend to mimicking the artist style of an era or genre.

Since PseudoClient effectively learns a model of one's personal style preferences, its feedback capability can also be repurposed for personal authentication. Google's reCAPTCHA system serves hundreds of millions of CAPTCHAs every day to tell humans apart from computers [53]. Users are presented a set of 9 or 16 square images and asked to identify which images contain certain objects. For our use case, we can first ask a user to select a small positive support set of design images upon the creation of an account. PseudoClient may then serve as an authentication system, in similar fashion to reCAPTCHA, by asking the user to select the design images that most align with their personal style preferences and judging its overall stylistic similarity with the positive support set associated with the account for authentication. The core assumption behind is that one's unique style preference can be personal enough to serve as a *mental metric* (as opposed to biometrics) for personal identification.

6.3 Generation

PseudoClient may also serve as a key component in generative design methods, such as Generative Adversarial Networks (GANs) [21]. We can think of GANs as two actors, a generator and a discriminator, working against each other to generate realistic designs. The generator attempts to hallucinate "fake" yet plausible designs. The discriminator attempts to differentiate between "fake" designs, generated by the generator, and "real" designs, from a set of real design examples. Competing against each other, each actor becomes better and better at doing its job, until the generated "fake" design is barely distinguishable from a "real" design. PseudoClient may be used as a discriminator (Figure 7c). Instead of differentiating between "real" and "fake" designs, our new discriminator differentiates between designs that the client "likes" or "dislikes" with respect to their personal style preference. On the other hand, the generator (or "PseudoDesigner") attempts to minimize the difference between its generated designs and the "like" examples provided by the client. Designers may thus utilize this adversarial behavior to synthesize designs that resemble the client's personal style preferences. We hope that such a generative system can become a useful tool, not to replace the role of designers, but to assist designers in serving as a source of inspiration, for rapidly prototyping new alternatives, and fundamentally making design work more accessible to novices.

7 LIMITATIONS AND FUTURE WORK

While PseudoClient performs well and appears to learn personal style, there are several limitations. These limitations suggest interesting avenues for future work. First, while PseudoClient is able to learn personal style holistically, more fine-grained elements such as variations in typeface may be overlooked, perhaps due to downsampling. A possible approach to address this may be to first semantically segment a design in order to distinguish between different design elements, as opposed to treating the entire design as a whole. For example, a design's typeface or background (with foreground subtracted) could be treated as separate features for learning. This being said, adding more feature engineering could lead to overly constrained systems. Second, a limitation raised by a fellow designer is that while PseudoClient is able to learn style in a *visual* sense, it is not able to explicitly understand a design based its *content*. For example, the use of skulls and bones in Figure 6b may correlate with specific styles (e.g., grunge, vintage, spooky), although their usage per individual may be different based on how they interpret these symbols and the context that surrounds them. It would be interesting to explore how content could correlate to personal elicitations of style and how (in)consistent they might be across different individuals. Third, while our evidence suggests that PseudoClient is able to distinguish between different styles, we may see that the positive and negative styles of participants in our experiments were quite different from each other. This motivates future work on exploring various degrees of similarity between positive and negative styles, such as investigating how PseudoClient can learn very subtle style differences. Finally, PseudoClient currently functions more as a component rather than as a fully fleshed out design application. For future work, we plan to work with PseudoClient as a design material [57] to implement some of the design tools discussed in the Applications section and evaluate them via user studies with designers and clients.

8 POTENTIAL RISKS AND IMPLICATIONS

The introduction of PseudoClient into the design workflow may also come with potential risks and implications. For example, due to the nature of black box models, designers may not be able to interpret and explain the suggestions made by PseudoClient to their clients. Arguably, clients may feel that the design decisions are purely arbitrary. An extension to increase interpretability may be to generate an activation map [43] that visualizes which regions of the design the model focuses on for making its predictions. Another potential risk is that our system when utilized in a search system could bury certain artists' work due to potential biases in the network. Future work should explore whether different decisions in the modeling process may lead to some kinds of work not being surfaced. Ultimately, as with any assistive system in a real world deployment, one should consider how the system affects various stakeholders and be wary of being overly reliant on the system's suggestions.

9 CONCLUSION

This paper demonstrates how we can leverage the pattern recognition capability of computational models to learn personal style preferences from only a few examples. We offer a set of principles built on prior work to ground our solution. Based on these principles, we designed PseudoClient, a metric-learning-based deep learning framework that learns a model of personal graphic design style. PseudoClient operates in an example-based manner, without relying on subjective style terms, and requires only a small handful of examples. In various experiments, we demonstrate that PseudoClient outperforms several alternative methods and offer an understanding of the levers that designers can alter to adjust or improve PseudoClient's ability in learning personal style. Finally, we discuss several applications that could be powered by PseudoClient from three directions: search, feedback, and generation.

This work takes a step towards the philosophy of computational design understanding with only a small number of data samples, which we argue increases the practicality of machine learning methods for design applications. We hope PseudoClient can be used as a building block to support the development of future design applications and serve as a foundation for future work on computationally understanding visual design style. For more information, please visit https://chuanenlin.com/personalstyle.

ACKNOWLEDGMENTS

We thank all participants who selected images for our experiments and the anonymous reviewers for their comments We also thank Zach Rupert, Tatooine Girl, Ivan Dubovik, Justin Bryant, Aleksandr Reva, Maycon Prasniewski, Hanna Ak, Tyler Thorny, Tatak Waskitho, Kane Young, Rick Barker, Paulius Kolodzeiskis, Anna Shulha, Vee Are, Evan Brown, and Jeff Trish for their beautiful illustrations. This work is supported by a gift from Accenture Technology Labs.

REFERENCES

[1] 2021. *Being a Copy Cat Is a Good Thing.* Retrieved February 1, 2021 from https://www.artistsnetwork.com/art-mediums/drawing/being-a-drawing-copy-cat-is-a-good-thing
[2] 2021. *Brand Deck.* Retrieved February 1, 2021 from https://branding.cards
[3] 2021. *OpenCV: Histograms - OpenCV documentation.* Retrieved February 1, 2021 from https://docs.opencv.org/3.4/d6/dc7/group__imgproc__hist.html
[4] Jennifer L. Aaker. 1997. Dimensions of Brand Personality. *Journal of Marketing Research* 34, 3 (1997), 347–356. https://doi.org/10.2307/3151897
[5] Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Trans. on Knowl. and Data Eng.* 17, 6 (June 2005), 734–749. https://doi.org/10.1109/TKDE.2005.99
[6] Sean Bell and Kavita Bala. 2015. Learning Visual Similarity for Product Design with Convolutional Neural Networks. *ACM Trans. Graph.* 34, 4, Article 98 (July 2015), 10 pages. https://doi.org/10.1145/2766959
[7] Zoya Bylinskii, Nam Wook Kim, Peter O'Donovan, Sami Alsheikh, Spandan Madan, Hanspeter Pfister, Fredo Durand, Bryan Russell, and Aaron Hertzmann. 2017. Learning Visual Importance for Graphic Designs and Data Visualizations. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (Québec City, QC, Canada) *(UIST '17)*. Association for Computing Machinery, New York, NY, USA, 57–69. https://doi.org/10.1145/3126594.3126653

[8] Chunyang Chen, Sidong Feng, Zhengyang Liu, Zhenchang Xing, and Shengdong Zhao. 2020. From Lost to Found: Discover Missing UI Design Semantics through Recovering Missing Tags. *Proc. ACM Hum.-Comput. Interact.* 4, CSCW2, Article 123 (Oct. 2020), 22 pages. https://doi.org/10.1145/3415194

[9] Peggy Chi, Zheng Sun, Katrina Panovich, and Irfan Essa. 2020. Automatic Video Creation From a Web Page. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '20)*. Association for Computing Machinery, New York, NY, USA, 279–292. https://doi.org/10.1145/3379337.3415814

[10] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z. Wang. 2006. Studying Aesthetics in Photographic Images Using a Computational Approach. In *Proceedings of the 9th European Conference on Computer Vision - Volume Part III* (Graz, Austria) *(ECCV'06)*. Springer-Verlag, Berlin, Heidelberg, 288–301. https://doi.org/10.1007/11744078_23

[11] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z. Wang. 2008. Image Retrieval: Ideas, Influences, and Trends of the New Age. *ACM Comput. Surv.* 40, 2, Article 5 (May 2008), 60 pages. https://doi.org/10.1145/1348246.1348248

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei Fei Li. 2009. ImageNet: a Large-Scale Hierarchical Image Database. *IEEE Conference on Computer Vision and Pattern Recognition*, 248–255. https://doi.org/10.1109/CVPR.2009.5206848

[13] Steven P. Dow, Kate Heddleston, and Scott R. Klemmer. 2009. The Efficacy of Prototyping under Time Constraints. In *Proceedings of the Seventh ACM Conference on Creativity and Cognition* (Berkeley, California, USA) *(C&C '09)*. Association for Computing Machinery, New York, NY, USA, 165–174. https://doi.org/10.1145/1640233.1640260

[14] Aaron Draplin. 2016. *Draplin Design Co.: Pretty Much Everything.* Harry N. Abrams.

[15] James Elkins. 2012. *Art Critiques: A Guide.* New Academia Publishing.

[16] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2019. Neural Architecture Search: A Survey. *Journal of Machine Learning Research* 20, 55 (2019), 1–21.

[17] Li Fei-Fei, Rob Fergus, and Pietro Perona. 2006. One-Shot Learning of Object Categories. *IEEE Trans. Pattern Anal. Mach. Intell.* 28, 4 (April 2006), 594–611. https://doi.org/10.1109/TPAMI.2006.79

[18] Elena Garces, Aseem Agarwala, Diego Gutierrez, and Aaron Hertzmann. 2014. A Similarity Measure for Illustration Style. *ACM Trans. Graph.* 33, 4, Article 93 (July 2014), 9 pages. https://doi.org/10.1145/2601097.2601131

[19] Dedre Gentner, Jeffrey Loewenstein, and Leigh Thompson. 2004. Learning and Transfer: A General Role for Analogical Encoding. *J Educ Psychol* 95 (03 2004). https://doi.org/10.1037/0022-0663.95.2.393

[20] Mary L Gick and Keith J Holyoak. 1983. Schema Induction and Analogical Transfer. *Cognitive psychology* 15, 1 (1983), 1–38.

[21] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, Vol. 27. 2672–2680. https://doi.org/10.1145/3422622

[22] James Hafner, Harpreet S. Sawhney, William Equitz, Myron Flickner, and Wayne Niblack. 1995. Efficient Color Histogram Indexing for Quadratic Form Distance Functions. *IEEE transactions on pattern analysis and machine intelligence* 17, 7 (1995), 729–736.

[23] Scarlett R. Herring, Chia-Chen Chang, Jesse Krantzler, and Brian P. Bailey. 2009. Getting Inspired! Understanding How and Why Examples Are Used in Creative Design Practice. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Boston, MA, USA) *(CHI '09)*. Association for Computing Machinery, New York, NY, USA, 87–96. https://doi.org/10.1145/1518701.1518717

[24] Forrest Huang, John F. Canny, and Jeffrey Nichols. 2019. Swire: Sketch-Based User Interface Retrieval. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland UK) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–10. https://doi.org/10.1145/3290605.3300334

[25] Phillip Isola, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. 2011. What makes an image memorable? *Journal of Vision* 11, 145–152. https://doi.org/10.1109/CVPR.2011.5995721

[26] Sergey Karayev, Matthew Trentacoste, Helen Han, Aseem Agarwala, Trevor Darrell, Aaron Hertzmann, and Holger Winnemoeller. 2014. Recognizing Image Style. In *Proceedings of the British Machine Vision Conference.* BMVA Press. https://doi.org/10.5244/C.28.122

[27] Gabriella Kazai, Jaap Kamps, and Natasa Milic-Frayling. 2013. An analysis of human factors and label accuracy in crowdsourcing relevance judgments. *Information retrieval* 16, 2 (2013), 138–178.

[28] Diederik Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations* (12 2014).

[29] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, Vol. 2. Lille.

[30] Brian Kulis. 2012. Metric Learning: A Survey. *Foundations and Trends in Machine Learning* 5 (01 2012). https://doi.org/10.1561/2200000019

[31] Ranjitha Kumar, Arvind Satyanarayan, Cesar Torres, Maxine Lim, Salman Ahmad, Scott R. Klemmer, and Jerry O. Talton. 2013. *Webzeitgeist: Design Mining the Web.* Association for Computing Machinery, New York, NY, USA, 3083–3092. https://doi.org/10.1145/2470654.2466420

[32] Manuel Lagunas, Elena Garces, and Diego Gutiérrez. 2019. Learning Icons Appearance Similarity. *Multimedia Tools and Applications* (04 2019). https://doi.org/10.1007/s11042-018-6628-7

[33] Tianqiang Liu, Aaron Hertzmann, Wilmot Li, and Thomas Funkhouser. 2015. Style Compatibility for 3D Furniture Models. *ACM Trans. Graph.* 34, 4, Article 85 (July 2015), 9 pages. https://doi.org/10.1145/2766898

[34] Andrés Lucero. 2012. Framing, Aligning, Paradoxing, Abstracting, and Directing: How Design Mood Boards Work. In *Proceedings of the Designing Interactive Systems Conference* (Newcastle Upon Tyne, United Kingdom) *(DIS '12)*. Association for Computing Machinery, New York, NY, USA, 438–447. https://doi.org/10.1145/2317956.2318021

[35] Iaroslav Melekhov, Juho Kannala, and Esa Rahtu. 2016. Siamese network features for image matching. 378–383. https://doi.org/10.1109/ICPR.2016.7899663

[36] Eleni Michailidou, Simon Harper, and Sean Bechhofer. 2008. Visual Complexity and Aesthetic Perception of Web Pages. In *Proceedings of the 26th Annual ACM International Conference on Design of Communication* (Lisbon, Portugal) *(SIGDOC '08)*. Association for Computing Machinery, New York, NY, USA, 215–224. https://doi.org/10.1145/1456536.1456581

[37] Naila Murray, Luca Marchesotti, and Florent Perronnin. 2012. AVA: A large-scale database for aesthetic visual analysis. *Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2408–2415. https://doi.org/10.1109/CVPR.2012.6247954

[38] Peter O'Donovan, Jundefinednis Lundefinedbeks, Aseem Agarwala, and Aaron Hertzmann. 2014. Exploratory Font Selection Using Crowdsourced Attributes. *ACM Trans. Graph.* 33, 4, Article 92 (July 2014), 9 pages. https://doi.org/10.1145/2601097.2601110

[39] Sinno Jialin Pan and Qiang Yang. 2010. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering* 22, 10 (2010), 1345–1359. https://doi.org/10.1109/TKDE.2009.191

[40] Katharina Reinecke, Tom Yeh, Luke Miratrix, Rahmatri Mardiko, Yuechen Zhao, Jenny Liu, and Krzysztof Z. Gajos. 2013. *Predicting Users' First Impressions of Website Aesthetics with a Quantification of Perceived Visual Complexity and Colorfulness.* Association for Computing Machinery, New York, NY, USA, 2049–2058. https://doi.org/10.1145/2470654.2481281

[41] Daniel Ritchie, Ankita Arvind Kejriwal, and Scott R. Klemmer. 2011. D.Tour: Style-Based Exploration of Design Example Galleries. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (Santa Barbara, California, USA) *(UIST '11)*. Association for Computing Machinery, New York, NY, USA, 165–174. https://doi.org/10.1145/2047196.2047216

[42] Babak Saleh, Mira Dontcheva, Aaron Hertzmann, and Zhicheng Liu. 2015. Learning Style Similarity for Searching Infographics. In *Proceedings of the 41st Graphics Interface Conference* (Halifax, Nova Scotia, Canada) *(GI '15)*. Canadian Information Processing Society, CAN, 59–64.

[43] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-CAM: Visual Explanations From Deep Networks via Gradient-Based Localization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV).* https://doi.org/10.1007/s11263-019-01228-7

[44] Dawn Shaikh and Barbara Chaparro. 2016. *Perception of Fonts: Perceived Personality Traits and Appropriate Uses.* 226–247. https://doi.org/10.1142/9789814759540_0013

[45] Lior Shamir, Tomasz Macura, Nikita Orlov, D. Mark Eckley, and Ilya G. Goldberg. 2010. Impressionism, Expressionism, Surrealism: Automated Recognition of Painters and Schools of Art. *ACM Trans. Appl. Percept.* 7, 2, Article 8 (Feb. 2010), 17 pages. https://doi.org/10.1145/1670671.1670672

[46] Martin Siefkes and Emanuele Arielli. 2018. *The Aesthetics and Multimodality of Style: Experimental Research on the Edge of Theory.* Peter Lang.

[47] Edgar Simo-Serra and Hiroshi Ishikawa. 2016. Fashion Style in 128 Floats: Joint Ranking and Classification using Weak Data for Feature Extraction. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR).* https://doi.org/10.1109/CVPR.2016.39

[48] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.* http://arxiv.org/abs/1409.1556

[49] Moeko Takagi, Edgar Simo-Serra, Satoshi Iizuka, and Hiroshi Ishikawa. 2017. What makes a style: Experimental analysis of fashion prediction. In *Proceedings of the IEEE International Conference on Computer Vision Workshops.* 2247–2253. https://doi.org/10.1109/ICCVW.2017.263

[50] Hossein Talebi and Peyman Milanfar. 2017. NIMA: Neural Image Assessment. *IEEE Transactions on Image Processing* PP (09 2017). https://doi.org/10.1109/TIP.2018.2831899

[51] Kristen Vaccaro, Sunaya Shivakumar, Ziqiao Ding, Karrie Karahalios, and Ranjitha Kumar. 2016. The Elements of Fashion Style. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (Tokyo, Japan) *(UIST '16)*. Association for Computing Machinery, New York, NY, USA, 777–785. https:

//doi.org/10.1145/2984511.2984573

[52] Andreas Veit, Balazs Kovacs, Sean Bell, Julian McAuley, Kavita Bala, and Serge Belongie. 2015. Learning Visual Clothing Style With Heterogeneous Dyadic Co-Occurrences. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[53] Luis Von Ahn, Manuel Blum, Nicholas J Hopper, and John Langford. 2003. CAPTCHA: Using Hard AI Problems for Security. In *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 294–311.

[54] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. 2014. Learning Fine-grained Image Similarity with Deep Ranking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[55] Ziming Wu, Taewook Kim, Quan Li, and Xiaojuan Ma. 2019. Understanding and Modeling User-Perceived Brand Personality from Mobile Application UIs. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland UK) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300443

[56] Anbang Xu, Shih-Wen Huang, and Brian Bailey. 2014. Voyant: Generating Structured Feedback on Visual Designs Using a Crowd of Non-Experts. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing* (Baltimore, Maryland, USA) *(CSCW '14)*. Association for Computing Machinery, New York, NY, USA, 1433–1444. https://doi.org/10.1145/2531602.2531604

[57] Qian Yang. 2020. *Profiling Artificial Intelligence as a Material for User Experience Design.* Ph.D. Dissertation. Carnegie Mellon University.

[58] Matthew D Zeiler and Rob Fergus. 2014. Visualizing and Understanding Convolutional Networks. In *European Conference on Computer Vision*. Springer, 818–833.

[59] Nanxuan Zhao, Ying Cao, and Rynson W. H. Lau. 2018. What Characterizes Personalities of Graphic Designs? *ACM Trans. Graph.* 37, 4, Article 116 (July 2018), 15 pages. https://doi.org/10.1145/3197517.3201355

[60] Xianjun Sam Zheng, Ishani Chakraborty, James Jeng-Weei Lin, and Robert Rauschenberger. 2009. Correlating Low-Level Image Statistics with Users - Rapid Aesthetic and Affective Judgments of Web Pages. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Boston, MA, USA) *(CHI '09)*. Association for Computing Machinery, New York, NY, USA, 1–10. https://doi.org/10.1145/1518701.1518703