BEING CREATIVE: A CROSS-DOMAIN MAPPING NETWORK

Jichen Wu

Graduation Thesis,   December 2019

Supervised by
Maarten H. Lamers & **Wojtek Kowalczyk**

Media Technology M.Sc. program, Leiden University

ABSTRACT

The ability of feature extractions from object/concept and feature connections is believed to be crucial to creativity. This paper proposed a novel method to learn extractions and connections separately. Such separation enables adaptive feature and object connections, which means one object can be connected to different other objects within different contexts. It is an attempt to apply recent cognitive theories of computational creativity in an actual computer. It also suggests a possibility that deep networks for image-to-image translations can be used to design highly automatic creative cognition models.

1 Introduction

It is not easy to answer the question: what is creativity? Nevertheless, many agree that the ability to make connections or combinations between concepts is a key component of creativity. Sternberg[1] suggested three processes for creativity: selective encoding, combination and comparison. Spelke et al.[2] stated that human cognition makes the assumption that the world is represented by objects and relations. Mekern et al.[3] also pointed out this agreement and implied a unitary model of creative cognition. This model, which is inspired by [4], is based on adaptive relations between features of concepts and ideas (so-called event files). "Features of concepts" means that the relations are not directly made between concepts but through features and one concept is represented by many independent features. "Adaptive" implies the weighting of possible features according to different situations so that the contributions of different features vary to adapt to the current situation. This unitary creative model is believed to be more integrative and complete than earlier models such as the dual-process model[5].

On the practical end, however, to the best of my knowledge, few relational models have been built in an artificial machine with an "end-to-end" learning fashion (which is largely adopted in the field of machine learning due to cheap data). Some previous studies did model the features of concepts and relations between them. Olecteanu et al.[6] used an "Object Replacement and Composition" system, in which the features of an object seem to have to be manually decided and contain a limited number of objects. This approach makes the dataset accurate and human-understandable, however, as they pointed out, the system needs a larger dataset or a different approach to build the feature space in order to perform on a larger scope. In another creative machine[5], features of an image object are extracted directly based on color and texture information. Although this approach avoided manually building datasets, it took the risk of losing much information, for example, distribution of color, shape and deeper meaning of the image. These issues revealed the need for a more general method to efficiently acquire more comprehensive features and relations.

How to efficiently extract features from raw data and build relations between them in an artificial system? The development of neural networks makes it possible to extract various kinds of useful information from big data. In particular, Self-organizing Map[7], Restricted Boltzmann Machine[8] and Autoencoder[9][10] are typical for dimensionality reduction. The reduced dimensions represent compressed information about the original data. Since it is representative, this compressed information, in a way, can be understood as features of the original data.

Assuming that these features are of the same nature as those proposed by[3][4][5] [6], the problem is stated as follows: Given two domains X and Y, we extract features from points in these domains into feature spaces Z_X and Z_Y. This problem is how to find various mapping functions to map Z_X onto Z_Y so that different mapping functions are selected under different situations. Many previous researches [5] [6] [11] [12] can only find a single mapping function because it relies on some level of equality between the feature vectors. For example, a red, thick T-shirt should be mapped to red, thick trousers. However, humans do not only relate two things by their equalities. For example, we might think a red, thin T-shirt makes good pairs with blue, thick trousers because of influence of fashion trends or because they are similarly rare in their domains. To find multiple mapping functions I propose two criterias. First, if humans experience two things together for several times, they may naturally connect them. So if anteriorly experienced pairs from X and Y are provided, a mapping function should map their feature vectors together. This criteria is named: Anteriorly experienced mapping. Second, similar objects in domain X should be mapped to similar objects in domain Y, while much dissimilar objects in X should be mapped to much dissimilar objects in Y. This criteria is named: Topology mapping.

While [11][13] can already learn non-deterministic mappings between domains, they still cannot learn different mappings based on different criterias, rather the mapped data points are selected based on statistic distributions. This research intends to build adaptive mappings between domains that are subject to (simulated) environmental or mindset changes. This research serves as a computational model for the recent creativity theories. It aims to make contributions to bridging the cognitive theory that creativity is facilitated by making connections between features of concept [3] and the computational applications that make connections between domains of data. If it is possible to build a model that can efficiently construct features and relations from raw data, on one hand, researches on implementing cognition inspired creative models in artificial machines, like [6] and [5], may have a powerful alternative to build computational models and result in having a better understanding of the cognitive models. On the other hand, pure statistic-driven computational applications could get theoretical inspirations and be more creative.

The rest of the paper is organized as follows. First, related work regarding the use of features in computational creativity theories and its relevance in computational technologies is introduced in section 2. The methodology and framework of the Cross-Domain Mapping Network (CDMN) are described in section 3. In section 4, the experiments regarding the effectiveness and creative behaviours of CDMN is carried out. Finally, conclusion is provided in section 5.

2 Related Work

This section is divided into 4 parts. First, the recent trend of considering feature in creative cognition, specifically, feature encoding and connection, is introduced in section 2.1. Machine learning methods for feature encoding are briefly described in section 2.2. In section 2.3, methods for feature connection are recognized in the field of machine learning, specifically, image-to-image mapping networks. Recent attempts to generalize these methods for multi-modalities (audio, image, video) are shown in section 2.4

2.1 Features and Computational Creativity

A feature set of a concept is distributed representations of a concept. The "distributed" means such a feature set are fully representative of this concept on a certain level of importance. Feature extractions and feature connections have an important role in the theory of computational creativities. Since a distinction was made between convergent thinking and divergent thinking[14], the studies of both thinking processes gradually revealed the role of features in creative thinking.

Divergent thinking is to generate creative ideas or solving problems creatively by exploring many possible solutions. It is identified to be associative [15][16] such that items stored in memory are associated or combined under a specific context[17]. Kenett et al.[18] found that highly creative individuals find stronger links between different components that have similar features. and are able to move further through the (semantic) network within a limited number of associations. To understand the mechanism of divergent thinking, Olteanu et al.[6] pointed out that if we store all associations between items in our memory, the combinatorial explosion will occur since everything could be potentially associated with anything else. To explain this, a model is built, consisting of three levels: problem template level, concept level and feature level. All concepts and objects are encoded in the memory with their features and the replacement of an object or concept is done by finding correlated features of different objects under a different context. For example, a DVD spindle case can be seen as a cup because they are both concave and can contain water. However, as pointed out in section 1, the implementation of such a model is inefficient because it requires to manually encode concepts into features.

Convergent thinking is to give a single best solution to a well-defined question. It is characterised by the Remote Associate Test (RAT)[19], in which three words are given and a fourth word needs to be found that can be combined with each of the three words. Some RAT solvers [20] [21] and RAT problem generator[22] was developed with associations of word pairs modelled in the concept level. Although a feature level is not implemented in these models, it is important to notice that it is essential to avoid the combinatorial explosion in a realistic setting. Indeed, a biological spiking neuron model of RAT[23] suggests that words are stored by distributed representations[24][25]. That means a semantic concept is represented by a number of neurons, and each neuron participates in the representation of a number of concepts. This does not necessarily lead to the statement that the neurons operate at a feature level, but it indicates the similarities between such representations and the so-called features: a concept has a number of features and one feature is usually related to a number of concepts.

Both divergent and convergent thinkings are modelled explicitly in a dual-process computational painter by Augello et al[5]. This painter is based on the MicroPsi cognitive architecture[26] and replaces local parts of an image with novel substitutes. With features of images and image details stored in the memory, the divergent process finds associations between features while the convergent process controls the directions of and evaluates the associations. Besides the technique limitation that only color and texture features are used, such dual-process models are increasingly criticized because the outcomes of divergent and convergent processes are often hard to distinguish and outcomes of each process rely on some extent of the interplay between these two processes[3]

Seeing the separation and conflict between divergent and convergent thinkings as well as the importance of features to model creativity, Hommel et al.[4] and Merken et al.[3] proposed three essential ingredients for a unitary model for creative behaviour:
(1) Distributed representation of objects or concepts
(2) Contextualization of representations
(3) Individual differences
In other words, concepts are represented completely by distributed features such that a concept can be reconstructed by the features. Creative behaviours are facilitated by the interplay between features of different concepts. To achieve a degree of flexibility, under a different context, different connections between features are activated. The ability of contextualization also decides if an individual is more or less creative.

In this paper, I identify two processes from Merken's unitary model: (1) The encoding of distributed features (2) Flexible connections between features to facilitate contextualization and individual differences. In the remaining of this section, creative computational applications are reviewed by the extent to which they have realised these two processes. Along with these two processes, three criterias are proposed to evaluate a creative model: (1) Features are distributed and representative (2) The connections are flexible under different circumstances (3) Individual differences (flexible and persistent) can be modelled.

2.2 Encoding and Decoding Features

Although the importance of features is identified, the computational creativity community still does not have many reliable and automatic ways to encode (extract) features from concepts. Previous programs either involve manual encoding[6] or the encoded features are not well-distributed (a concept cannot be fully represented by its feature set)[5]. Fortunately, in the field of Neural Network, methods have been developed to encode features from raw data, including Restricted Boltzmann Machine[8] and Autoencoder[9][10]. Autoencoder has many variations that have different focus and deal with different type of data, for example, Denoising Autoencoder[27], Sparse Autoencoder[28], Contractive Autoencoder[29] and LSTM Autoencoder[30]. One may think that the features encoded by an autoencoder are just numbers and thus are of different nature as "color", "shape" or "texture". However, as we discussed in sub-section2.1, what makes features crucial is not whether they are abstract numbers, activation of neurons or concrete attributes but they serve as a set of distributed representations of a concept or an object. The encode-decode process of an autoencoder ensures the encoded features are fully representative of the input data.

One limitation of Autoencoder is that the encoded features are data-specific. For example, given a dataset of cups, the features will only capture the difference between a white cup and a blue cup, but never a white cup and a DVD spindle case. To have "concave" encoded in the features of cups and spindle cases, the dataset has to include many other objects besides cups and spindle cases so that "concave or not" becomes a significant attribute. Because of this, instead of the creative problem solving (as in [5] and [6]): seeing A as B, this paper will focus on creating B from A, which is also believed by Boden[31] to be combinatorial creativity and may also fall into the scope of exploratory creativity.

To achieve this the model needs not only "decode" the input back from the features, but also "generate" new data from arbitrarily given features. Such model is called generative model, for example, restricted Boltzmann machine [32] and variational autoencoder (VAE)[33]. These early methods have the problem that the generated data points trend to lose high-frequency features. In terms of image generating, the outputs are usually blurry[34]. After the invention of adversarial training[35], many are able to generate photo-realistic images [36][37] [38], videos[39] and music [40]. Although typical adversarial generative networks do not require an encoded feature space, it is possible to construct such feature space with an adversarial autoencoder [12].

2.3 Feature Connection

Many newest studies in the field of Domain Mapping Network (DMN) have already achieved the process of feature encoding and connection to some extent. However, because this field, derived from the field of Neural Network, has hardly been compared to creativity theory, such relevance is hardly mentioned or even realised. In section2.2, I identified the relevance of Neural Network in the encoding of features. Next, the relevance of DMN in the connecting of features will be identified.

DMN has constantly been studied in image-to-image translation. Early works [41][36][42][43], even though having the encode-decode structure in their generators, do not perform any manipulations on the encoded features, rather just encode features from an image from domain A and then directly decodes these features to generate an image in domain B. Some successors, including [13] and [44], found that having features as a condition enhanced the performance of the network. The key idea is that the features need to be consistent. That is, if an image b is generated from image a and we encode b again it should have the same features as a. However, in these methods, feature sets do not directly interact with each other. In other words, features from different domains are not explicitly connected.

It were Liu et al.[12] who proposed the "shared latent space assumption". Domain A and domain B have the same encoded feature space, so that the encoded features of image a, if fed into the decoder of domain A, will give back a and if fed into the decoder of domain B, will give back a new image b. It was the first work to model explicitly the connection of features, although this connection is simply features of a = features of b and there is no flexibility of such connections. Huang et al. [11] proposed an improvement of this model by adding a "not shared feature space" along with the "shared feature space". Although this gives some degree of flexibility of connections, it implies that some features are always connected and other features are never connected. While true flexibility is deciding which features are relevant to the context and should be connected adaptively. This paper will make an attempt to implement such flexibility.

2.4 Cross-modal Temporal Data Generation

A universal creativity model should be able to apply to data with different modality, rather than only image-to-image translation. The problem becomes more tricky if the domains are temporal data (etc. video, music) or require cross-modal (audio to visual, visual to audio) mapping. Two state-of-art music-to-music[45] and video-to-video[46] networks achieved good accuracy in their fields. However, their networks were specifically designed for their modalities which made it hard to generalize them to other modalities. Mor et al.[45] used the WaveNet structure[47], which takes music samples from all previous steps as input. If to be used with videos, WaveNet will post extreme demands on computational and memory requirements to process the complete previous video sequence. Wang et al.[46] used a network that takes several previous frames as input. However, to recognize the tempo and rhythm of audio or music, it needs to take tens or even hundreds of frames as input, which is also very expensive.

A few recent researches [48][49][50] focused on audio-to-image translation. These networks take the full audio piece as input, which means the audio is not treated as temporal data. However, for humans, we do not need to listen to the audio from the beginning to the end to start some visual imagination in our brain. Besides, it is more common to perceive constantly changing visuals with audio[51] or generate video with audio. Face generation from speech[52][53] and speech generation from silent video[54] can be defined as cross-modal temporal data generator. Different than these papers that focus on translation between face and speech, this work will focus on a more general situation: audio-video translation. Audio-video translation, together with image-image translation, will show that this Feature-Encoding-Feature-Connection model is general enough for different modalities.

3 Method

The two steps of creative cognition: Feature encoding and flexible feature connection, as we identified, are implemented in a computer program, called cross-domain mapping network, to show that such creative model can be described computationally. For feature connection, the implementation used mapping functions, as described in sub-section 3.1. For feature encoding and decoding, adversarial autoencoder networks are used and are described in sub-section 3.2. Most of these networks are specially designed for image data, expect the ones discussed in section 3.2.3, which is designed for temporal data. However, the principled methodology should be suitable for other kinds of data as well.

3.1 Mapping Functions

A mapping function is, in principle, a function that translates each data point z_x in feature space Z_X to a data point $z_{x\to y}$ in feature space Z_Y: $z_{x\to y} = m(z_x)$. Since, usually, there are infinite points in a feature space, the number of mapping functions are infinite. To attack this problem I first cluster Z_X and Z_Y into n_X and n_Y clusters and mapping functions map clusters together. Thus a mapping function from Z_X to Z_Y explore within the finite possibilities of size $(n_X)^{n_Y}$.

However much information could be lost if the feature vector only carries clustering information. To overcome this, a feature vector z_x is separated into two vectors, a cluster vector c_x in a finite space (a space that has finite points) and a vector carrying other detail information v_x in an infinite space. Next, the feature extraction function E_X is defined:

$$c_x, v_x = E_X(x)$$

Similarly we have E_Y. While there are $(n_X)^{n_Y}$ possible mappings from cluster vector space of X to cluster vector space of Y, v_x is passed unchanged: $v_{x\to y} = v_x$, just like [12]. The relation between c_x and v_x can be understood using figure 1. While c_x defines the center of a cluster in the space, v_x is a small vector deviating from this center. With number of clusters $n_x = n_y = 1$, this model is identical as the share latent space assumption proposed by [12]. With $n_x \to \infty$, $n_y \to \infty$ and $v \to 0$, in theory it is possible to find any arbitrary mapping functions. The whole process of mapping a data point x to domain Y is:

$$c_x, v_x = E_X(x)$$

$$c_{x\to y} = m(c_x)$$

$$y = G_Y(c_{x\to y}, v_x)$$

Here G_Y is a function that decode a feature vector back into the domain Y: $\hat{y} = G_Y(E_Y(y))$ where $\hat{y} \approx y$.

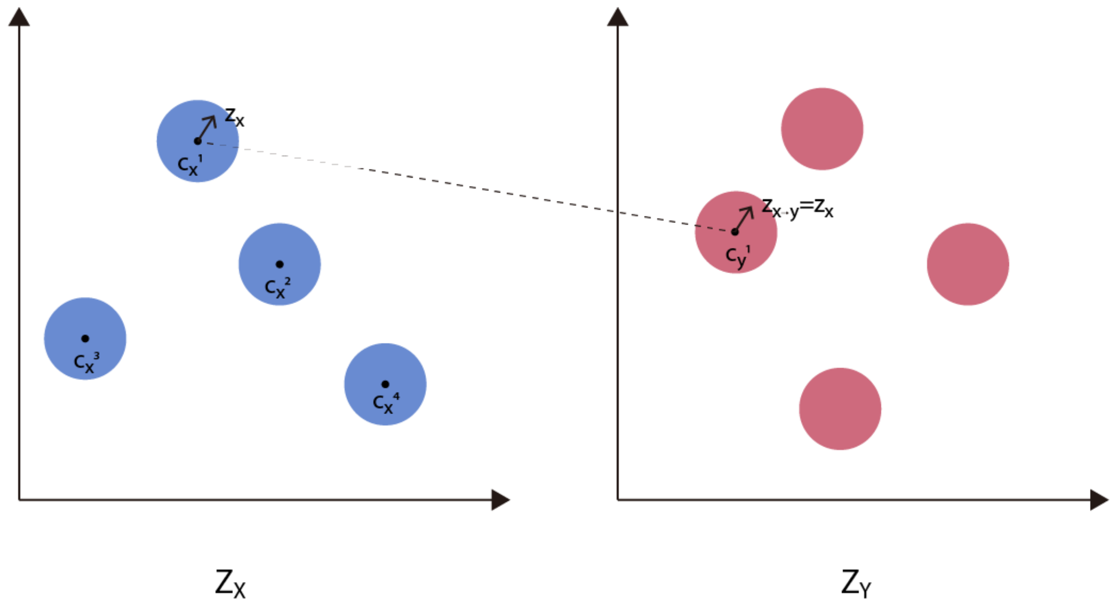

Figure 1: Relation between c and v and how the mapping function work

I use the help of two criteria: anteriorly experienced mapping and topology mapping, as mentioned in section 1, to find good mapping functions. First, when anteriorly experienced pairs are present: $\{(x_1, y_1), ..., (x_{nn}, y_{nn})\}$, assuming that $c_{x_i}, v_{x_i} = E_X(x_i)$, we get the paired clusters:

$$PC = \{(c_{x_i}, c_{y_i}) \quad i = 1, 2, ..., nn\}$$

Then, the loss of a mapping function can be measured by how good it matches PC:

$$\mathcal{L}_{pair} = \sum_{i=1}^{nn} w_i * eval(m(c_{x_i}), c_{y_i})$$

Where $eval(m(c_{x_i}), c_{y_i})$ returns 0 if $m(c_{x_i}) = c_{y_i}$, 1 otherwise. w_i is a weight assigned to each instance. By altering the weights $\{w_i, \quad i = \{1, 2, ..., nn\}\}$, each pair can have a different importance.

Second, when topology mapping is used, I first assume that the topology is preserved with the feature extraction functions E_X and E_Y so similar objects have small distance in feature space. The loss function measures how good a mapping function m preserve the distance between clusters of x:

$$\mathcal{L}_{topo} = \sum_{i=1}^{n_X} \sum_{j=1}^{n_X} (d(c_x^i, c_x^j) - d(m(c_x^i), m(c_x^j)))^2$$

where c_x^i represents the i-th cluster of the total n_X clusters and $d(c_x^i, c_x^j) \in [0, 1]$ is the normalized Euclidean distance between c_x^i and c_x^j. $\mathcal{L}_{topo}$ is called stress function in Multidimensional scaling.

Given the weights w_{topo}, the overall loss to be minimize is:

$$\mathcal{L}_{map} = \mathcal{L}_{pair} + w_{topo}\mathcal{L}_{topo}$$

An algorithm to find good solutions of $\mathcal{L}_{map}$ is subject to the criteria (see section 2.1): the mapping functions should be able to model individual differences. In this paper genetic algorithms are used because many design choices of them can facilitate the model of flexible and persistent (explorative and exploitative) individuals.

3.2 Network for Feature Encoding

The functions E_X, G_X, E_Y, G_Y are learned by a neural network. We assume that c_x, v_x, c_y, v_y are of the same dimensionality. Furthermore, c_x, c_y can be represented by one-hot vectors h_x, h_y:

$$c_x = H_X(h_x), \quad c_y = H_Y(h_y) \quad h_x \in \{0,1\}^{n_x}, h_y \in \{0,1\}^{n_y}, ||h_x||_2 = ||h_y||_2 = 1$$

We update E_X so that: $h_x, v_x = E_X(x)$ and update m so that $h_{x \to y} = m(h_x)$ (which does not change the functionality of E_X and m). The same change is also made to E_Y. Makhzani et al. [55] have shown that this way it is possible for the encoder to learn one-hot vectors representing clustering information.

The complete structure of the network is shown in figure 2. The network has two functions. When passing c_x, v_x to G_X, it is an autoencoder to reconstruct x, when passing $H_Y(m(h_x)), v_x$ to G_Y, it is a mapping network. For networks E and G, similar structures to [12] are used. E consists of only convolutional layers except the last layer to get h, which is fully connected, G consists of only convolutional layers and H is a fully connected layer.

3.2.1 Training for Autoencoding

The two autoencoder structures are trained independently. For autoencoder (E_X, H_X, G_X), we want the reconstructions $\hat{x}$ to be as similar as to the input x. Here L1 loss is used:

$$\mathcal{L}_{recon}^x = \mathbb{E}_{x \sim p(x)}[||x - \hat{x}||_1]$$

where $\hat{x} = G_X(H_X(h_x), v_x)$ and $h_x, v_x = E_X(x)$. v_x and v_y need to follow the same distribution for the mapping to work. We let them both follow distribution $N(0, I)$ where I is the identity matrix. This is done by using VAE structure[33] that uses KL-divergence loss:

$$\mathcal{L}_{KL}^x = KL(E_X(x)[1]||N(0, I))$$

where $E_X(x)[1] = v_x$ and h_x is expected to be a one-hot vector representing unsupervised clustering information. Adversarial training is used with a discriminator D_X that tries to tell h_x from a random real one-hot vector h_{real} of the

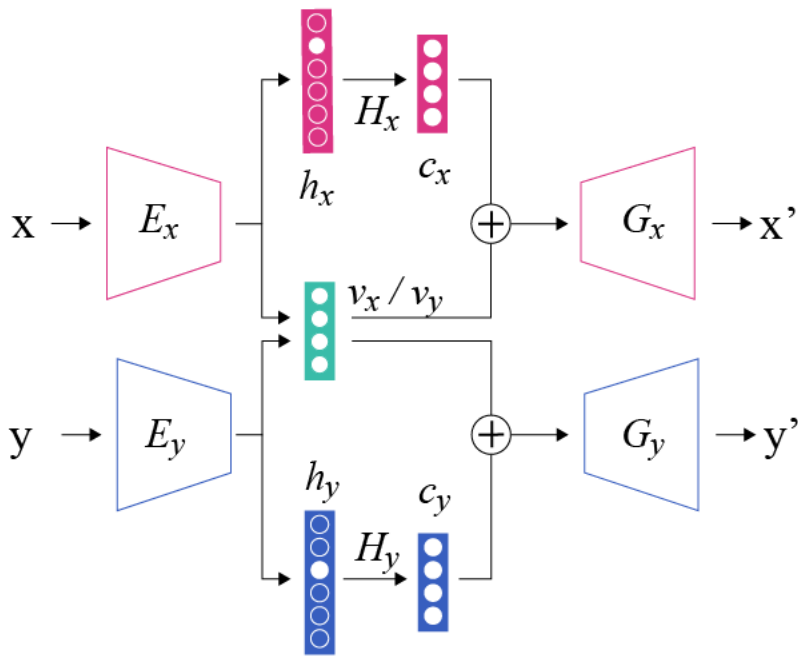

Figure 2: Complete structure of the domain-mapping network

same dimensionality:

$$\mathbf{L}^{x}_{adve} = \mathbb{E}_{x\sim p(x)}[log(1 - D_X(E_X(x)[0]))] + \mathbb{E}_{h_{real}\sim cat(n_x)}[log(D_x(h_{real}))]$$

where $E_X(x)[0] = h_x$ and $cat(n_x)$ is the distribution of all one-hot vectors with dimensionality n_x. It is also intuitive to make sure that any different cluster c_x has a distance from each other, so there is no overlap, like in figure 1. However, experiments in section 4 indicate that there is no effect in doing so. So here I do not present the loss to keep different clusters at a distance. An illustration of the training process is shown in figure 3

Similarly, the loss functions for y are:

$$\mathcal{L}^{y}_{recon} = \mathbb{E}_{y\sim p(y)}[||y - \hat{y}||_1]$$

$$\mathcal{L}^{y}_{KL} = KL(E_Y(y)[1]||N(0, I))$$

$$\mathcal{L}^{y}_{adve} = \mathbb{E}_{y\sim p(y)}[log(1 - D_Y(E_Y(y)[0]))] + \mathbb{E}_{h_{real}\sim cat(n_y)}[log(D_Y(h_{real}))]$$

The total loss is then:

$$\mathcal{L}^{ae}_{total} = \min_{E_X, E_Y, H_X, H_Y, G_X, G_Y} \max_{D_X, D_Y} w_{recon} * (\mathcal{L}^{x}_{recon} + \mathcal{L}^{y}_{recon}) + w_{kl} * (\mathcal{L}^{x}_{KL} + \mathcal{L}^{y}_{KL}) + w_{adve} * (\mathcal{L}^{x}_{adve} + \mathcal{L}^{y}_{adve})$$

3.2.2 Training for Mapping

There are two disadvantages if the network is only trained with L^{ae}_{total}. First, even though L_{KL} penalize v_x, v_y that do not follow Gaussian distribution, they tend to deviate from Gaussian distribution[55], especially in high dimensional space. This leads to the empirical observation that v_x and v_y follow different distribution and a mapping $G_Y(c_y, v_x)$ will only generate a noise output. Second, the clustering has strong bias. That is, there could be one cluster with half of the training set put into it and half of the clusters empty. To overcome these, a joint training process is designed. With v_x and a random vector h^r_y, $G_Y(H_Y(h^r_y), v_x)$ learns to generate a realistic image with the help of a discriminator D^{img}_Y

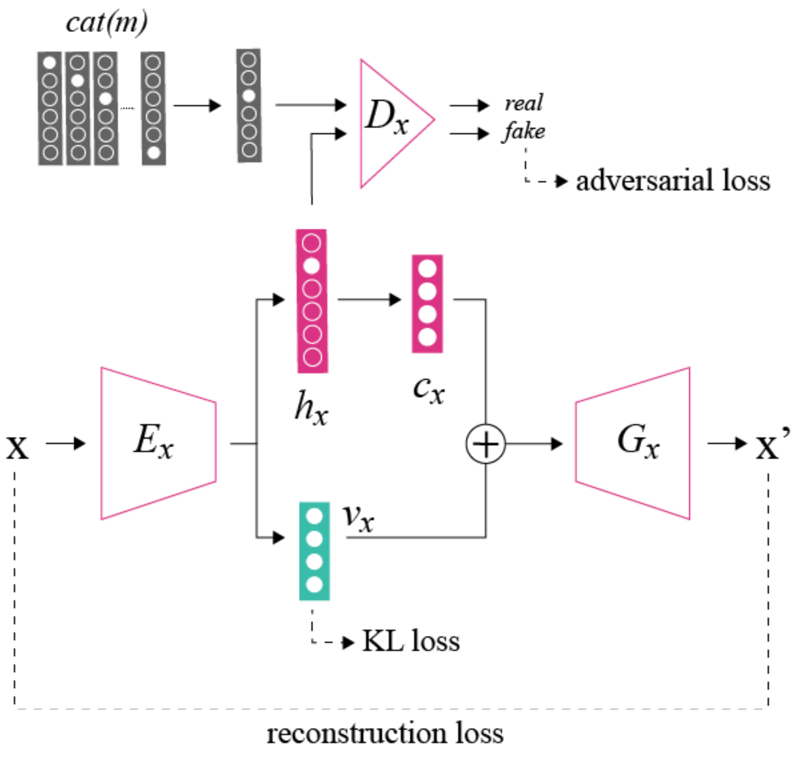

Figure 3: Network structure at training for autoencoding loss

for generated outputs:

$$\mathcal{L}^{y}_{GAN} = \mathbb{E}_{x\sim p(x),h^r_y\sim cat(n_y)}[log(1 - D^{img}_Y(G_Y(H_Y(h^r_y) + E_X(x)[1]))] + \mathbb{E}_{y\sim p(y)}[log(D^{img}_Y(y))]$$

This makes sure that G_Y learns the distribution of v_x and also the full distribution h_y. As several image-to-image translation networks suggested[43][12][13], cycle consistence loss is also used:

$$\mathcal{L}^{y}_{cyc} = \mathbb{E}_{x\sim p(x),h^r_y\sim cat(n_y)}[||v_x, h^r_y - E_Y(G_Y(H_Y(h^r_y) + v_x))||_1]$$

where $v_x = E_X(x)[1]$. An illustration of these process can be found in figure 4.

Similarly, the loss for $y \to x$ is:

$$\mathcal{L}^{x}_{GAN} = \mathbb{E}_{y\sim p(y),h^r_x\sim cat(m)}[log(1 - D^{img}_X(G_X(H_X(h^r_x) + E_Y(y)[1]))] + \mathbb{E}_{x\sim p(x)}[log(D^{img}_X(x))]$$

$$\mathcal{L}^{x}_{cyc} = \mathbb{E}_{y\sim p(y),h^r_x\sim cat(n_x)}[||v_y, h^r_x - E_X(G_X(H_X(h^r_x) + v_y))||_1]$$

The total loss is:

$$\mathcal{L}^{map}_{total} = \min_{E_X,E_Y,H_X,H_Y,G_X,G_Y} \max_{D^{img}_X,D^{img}_Y} w_{GAN} * (\mathcal{L}^{x}_{GAN} + \mathcal{L}^{y}_{GAN}) + w_{cyc} * (\mathcal{L}^{x}_{cyc} + \mathcal{L}^{y}_{cyc})$$

During the training process, the network is trained on $\mathcal{L}^{ae}_{total}$ and $\mathcal{L}^{map}_{total}$ iteratively.

3.2.3 Network for Cross-Modal Temporal Data

Besides the uni-modal image-to-image translation tasks, a creative model should also be able to solve cross-modal non-static (temporal) translation tasks. Such tasks post more restrictions on the autoencoder networks. A variant of

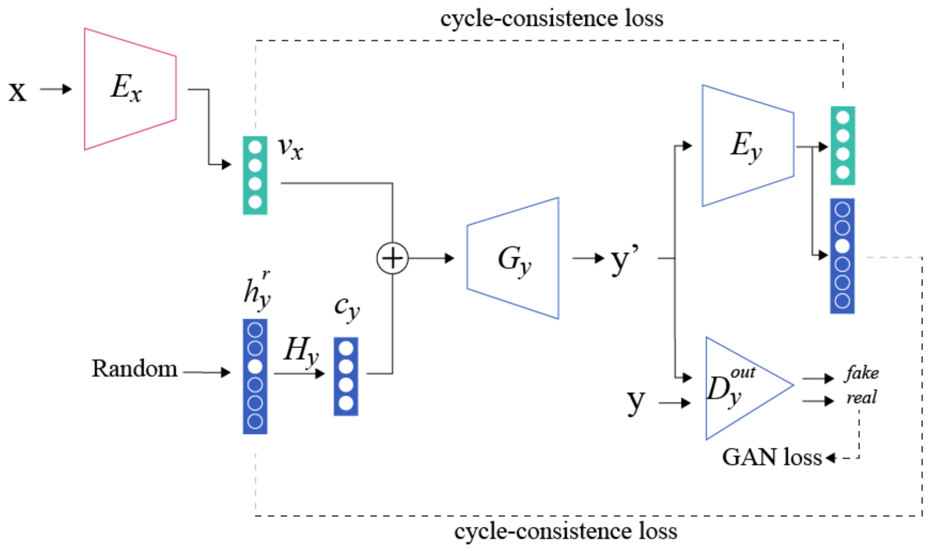

Figure 4: Network structure at training for mapping loss

the network shown in figure 2 is specially designed for audio-to-video translation. In this new version of the network, the functions E_X, E_Y, H_X, H_Y remain the same. Differences lays in G_X and G_Y where the layers which were all convolutional are replaced by LSTM convolutional blocks[56] and the new functions are named G_X^m and G_Y^m.

$$\hat{x}, mem_x^{t+1} = G_X^m(c_x, v_x, mem_x^t)$$

where mem_x^t is the memory of the network G_X^m at time t. At the beginning of each sequence, an empty memory mem_x^0 is fed into the network. An illustration of this network is shown in figure 5. Compared to other methods mentioned in section 2.4, the advantages of this network are: (1) LSTM blocks are able to "remember" data traced deep in the history with acceptable computational cost. This enable the recognition of tempo and rhythm of music; (2) This process of memorizing is believed to have some similarity to humans' memory process; (3) The features of a frame should be independent from recent frames. So instead of getting a feature set of a sequence of frames, the network get a feature set of one frame at a time.

4 Experiments

This section is divided into two parts. First, in section 4.1, the configurations and technical properties of the model is studied. Then, in section 4.2, scenarios are provided to study the model's human-like creative behaviour. To simplify to problem, throughout this section, there is no overlapping between mappings (two cluster of x cannot be mapped to the same cluster of y).

4.1 Performance Analysis

This section consist of studies of the configurations and technical properties of: (1) image-to-image mapping model in section 4.1.1 (2) music-to-video mapping model in section 4.1.2.

4.1.1 Image-to-image Mapping

In this section, the performance of the network is evaluated, fist quantitatively, then qualitatively.

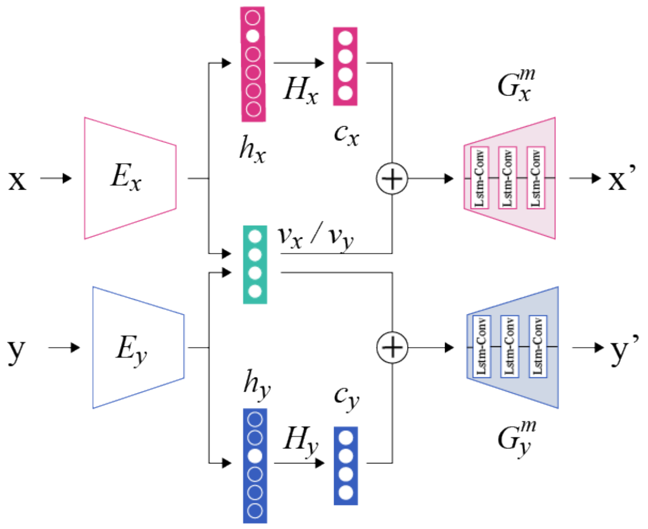

Figure 5: Structure of the network for audio-video mapping. Compare to figure 2, decoding functions G_x^m and G_y^m are now consisted of LSTM convolutional blocks instead of convolutional layers.

Quantitative Results

First, several sets of hyperparameters are evaluated. Several metrics are used for quantitative measurement. (1) The size of clusters is preferred to be balanced. Thus the standard deviation of the size of clusters is to be minimized(e_1); (2) Input images that are put into one cluster need to be similar with each others. Thus the mean of perceptual similarity[57] of input images is to be minimized(e_2); (3) It is noticed that when the cluster c is fixed and v varies, sometimes the network generates images that are almost identical. This means v does not carry valid information. To avoid this, fed with random v, generated images from the same cluster should be diverse. Thus the reciprocal of the mean of Euclidean distance between generated images is to be minimized(e_3).

The evaluation is performed using a clothing dataset from [58]. Blouses are used for domain X, trousers and skirt are used for domain Y, each image is cut by a window that only contains the piece of clothing and resized to 64*64. Batch normalization[59] is used throughout the encoders, as it is believed to be crucial for clustering[55]. Number of clusters $n_x = n_y = 50$. Tuned parameters are: weights of losses, whether or not to use layer normalization(LN)[60] in decoders, different target distribution of v and whether or not to spatially separate clusters. Here separated clusters mean that the distances between cluster centers are controlled so that there is no overlap between clusters. Suppose $v \sim N(0, I)$, $v \in \mathbb{R}^d$ and d is a large number, the distance between v and 0 is approximately $||v - 0||_2 = \sqrt{d}$. Thus the cluster centers are controlled:

$$||H_X(h_x^i) - H_X(h_x^j)||_2 > 2 * \sqrt{d} \quad \forall h_x^i, h_x^j, h_x^i \neq h_x^j$$

Similarly for h_y.

The results for several sets of hyperparameters are shown in table 1. Besides, it is also observed that using $N(0, 0.1 * I)$ and layer normalization greatly speed up the training. The effect of the separation of clusters is unclear. Since set 5 dominates set 9 with no separation, but set 1 also dominates set 8 with separation. LN and distribution of $N(0, I * 0.1)$ seem to achieve overall good results.

Then the first five set that forms the Pareto front are measured qualitatively and the results are shown in figure 6. Ideally, a generated image should share several important features with the input top image (color, shape) and it should not be too similar to other generated images from the same cluster. We can see although set 4 has the lowest e_2, the generated images hardly resemble inputs. While set 2 and set 5 cluster using more colors features, set 3 focuses more on the

Set index	Separate clusters	Distribution of v	Decoder normal	$w_{adve}, w_{reco}, w_{cyc}, w_{GAN}$	e_1	e_2	e_3
1	Yes	$N(0, I)$	None	10, 1, 0.1, 1	122.4	0.47	0.213
2	Yes	$N(0, I * 0.1)$	LN	10, 1, 0.1, 1	129.0	0.42	0.191
3	Yes	$N(0, I * 0.1)$	None	10, 1, 1, 1	**118.7**	0.48	0.203
4	Yes	$N(0, I * 0.1)$	LN	1, 10, 0.1, 0.1	127.2	**0.37**	0.318
5	No	$N(0, I * 0.1)$	LN	10, 1, 0.1, 1	125.5	0.42	**0.083**
6	Yes	$N(0, I)$	None	10, 1, 0.1, 1	131.5	0.48	1.072
7	Yes	$N(0, I)$	None	10, 1, 0.1, 0.1	135.2	0.47	0.819
8	No	$N(0, I)$	None	10, 1, 0.1, 1	128.9	0.47	0.485
9	Yes	$N(0, I)$	LN	10, 1, 0.1, 1	129.0	0.45	0.181

Table 1: quantitative results with different sets of hyperparameters. e_1: error for balanced cluster sizes to be minimized. e_2: error for input variety to be minimized. e_3: error for output diversity to be minimized. Sets 1-5 form the Pareto front

shapes. Also, although e_3 of set 5 is significantly lower than sets 2 and 3, perceptually their in-cluster varieties are about the same. In the following of this paper, if not specified, the hyperparameters of set 3 will be used. This choice is made mostly arbitrarily because none of the hyperparameter sets can be said to be superior than the rest from figure 6.

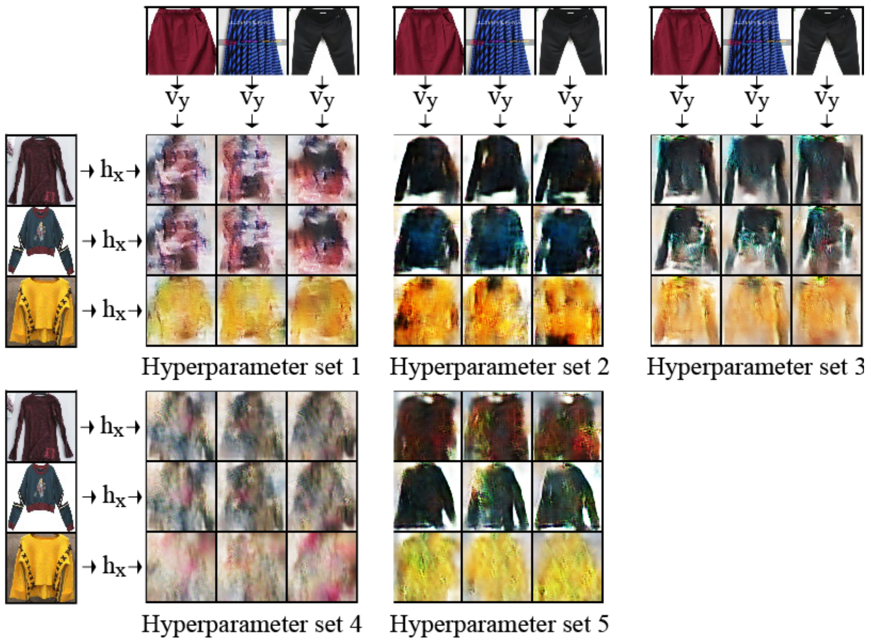

Figure 6: Comparison of 5 sets of the Pareto front (sets 1-5 from table 1). This is a matrix where v_y is extracted from the top row and h_x is extracted from the leftmost column. each element except these two lines are generated by G_x from the corresponding v_y and h_x

Qualitative results

First, the clustering property of the model is shown. Randomly selected test images from the same clusters are shown in figure 7, as well as the cluster center(If input is x, and reconstruction is $G_x(c_x + v_x)$, then cluster center is defined as $G_x(c_x + 0)$). It is noticed that more than half of the clusters are left empty.

A premise of the topology mapping method in section 3.1 to work is that the distance between clusters represents the similarity between the clusters, which is shown in the following. Here mnist dataset is used for both domain X and Y since it is easier to justify if two clusters of digits are similar to each other than some more complicated images. In

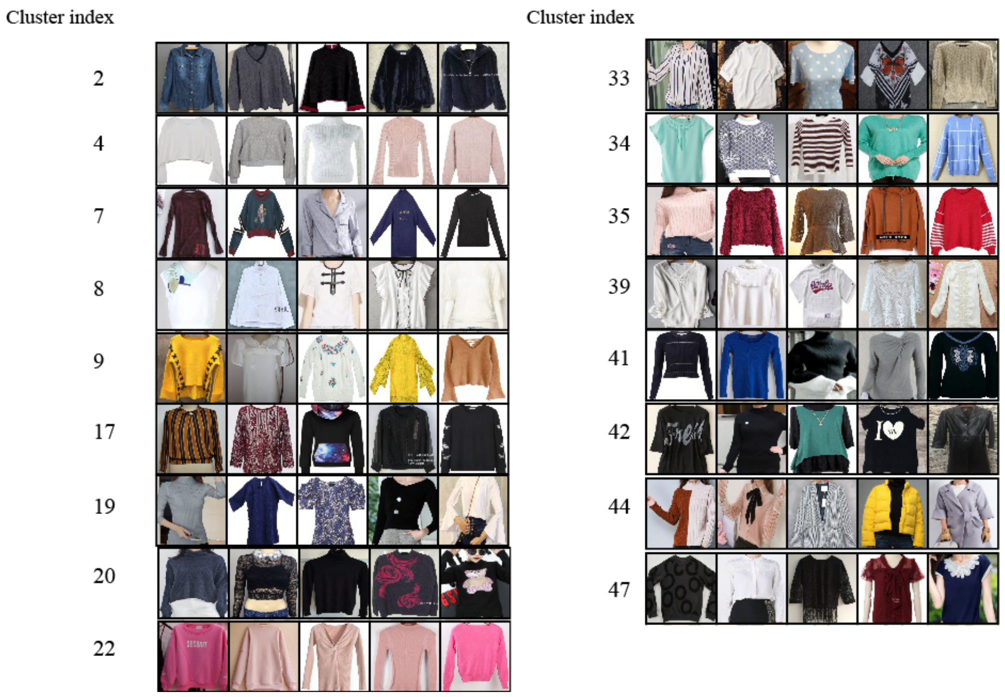

Figure 7: Random selected input that are put into the same clusters. Total number of clusters is 50.

figure 8 we can see that most closely related clusters have small distances in between. For example, two clusters of 1s (clusters 10 and 15), two clusters of 9s (clusters 14 and 16), cluster of 3s and clusters of 2 and 3s (clusters 5 and 12).

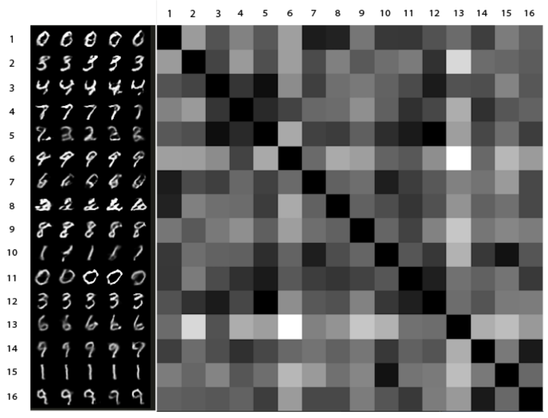

Figure 8: Left: 5 randomly sampled images from the same cluster each row. Right: confusion matrix of normalized distances between clusters. The darker the color is, the closer the two clusters. Number of clusters is 16

After identifying some key properties of the model, it is first tested on classical image-to-image translation datasets. These classical datasets are defined as that there is only one mapping rule that is human-interpretable. For example, in the edges2shoes dataset [43], it would only make sense to pair a shoe image with an edge image if the contour of the shoe image is the same as the edge image. Many researches[43][12][42][11] have already achieved good results on these datasets. Here I show that my network is also capable of learning this single mapping rule.

In figure 9, the results in edges2shoes dataset are shown. The network is first trained to minimize L^{ae}_{total} and L^{map}_{total}. Then, a single mapping function m_{deter} is learned from paired data. We can see that it is able to generate shoe images that have similar contours as the edge images. However, it is also limited in two aspects: 1. The contours of inputs and outputs do not match as well as other state-of-art networks [12][42]. 2. Compared to ground truth images, the colors of the outputs are limited. This might indicate that the network is unable to encode enough information into the vector v.

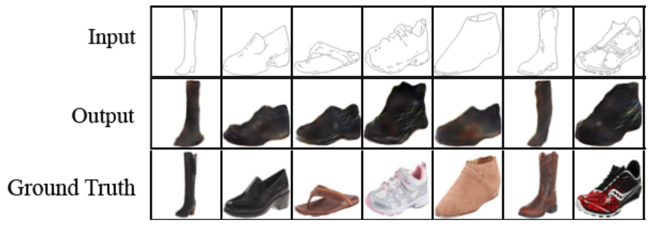

Figure 9: Edges to shoes dataset. Using prior pairs for mapping function

Next, the network is tested on a novel task: generate top clothing from bottom clothing. This is the scenario where multiple mapping rules are needed, because different people, in different situations, have different rules of how clothing should be paired. Here two rules are tested: 1. topology mapping. 2. color matching. The goal is to evaluate the Cross-Domain Mapping Network (CDMN)'s adaptivity to different rules. Adaptation to topology mapping is expected to generate similar images of top clothing when given similar images of bottom clothing. While adaptation to color matching is expected to generate top clothing that has the same color as the provided bottom clothing. To create pair images for color matching, for each bottom image, a main color is extracted by K-mean and a top image with the closest main color is selected. Dataset created in such a way is arbitrary because any other property except the main color is selected at random.

In figure 10, the topology mapping is compared with UNIT[12] and the color matching is compared with Pix2pix[42]. Comparing the result from UNIT[12] and rule 1 (topology mapping) of CDMN, which are both unsupervised, we can see that while UNIT tend to find a top that always corresponds to a given bottom (opposite color, similar texture and shape), CDMN using topology mapping of finds a structural relation that similar inputs always result in similar outputs without the correspondence of color, texture and shape between inputs and outputs. For rule 2 (color matching), CDMN is able to connect several major colors. It is noticed that Pix2pix is more accurate in color matching. However, it is also important to notice that when the mapping rule changes (for example, find bottom clothing that always has darker color), pix2pix has to learn it from the beginning, while CDMN only needs to find a new mapping function.

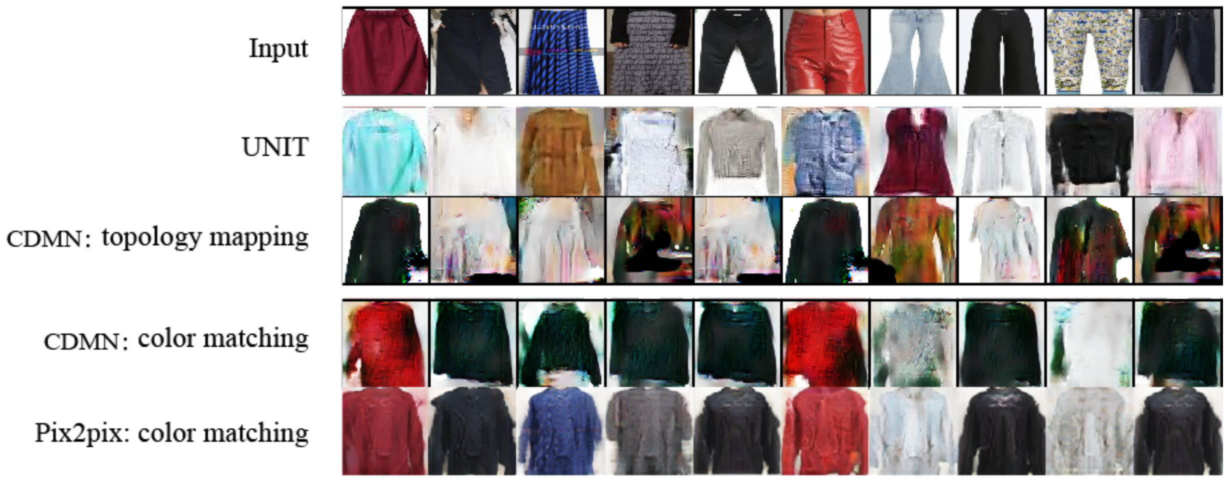

Figure 10: Comparison between outputs from different networks/ mapping functions.

4.1.2 Music Visualization

In this section the model's performance on music-to-video mapping is shown. In particular, I focus on video generation from music, but not the other direction. The reasons are, with knowledge of image generation, it is much easier to generate video than music and since the network is trained bidirectionally, even an unidirectional generation shows the model's behaviour bidirectionally.

The music dataset used throughout this paper is a selection of the Million Song Dataset[61], with 100 songs of 30 seconds long each. For the video dataset, a script is used to generate jumping circles with 10 videos of 200 frames each (an example is shown in figure 11).

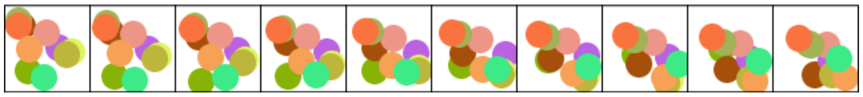

Figure 11: Ten successive frames of one video of the jumping circles dataset. The circles moving freely, do not collide with each other but get bounced at the boundaries. The ten videos consist of circles of different colors.

Quantitative Results

First, the impact of different hyperparameter sets is studied. I expect the behaviour of the music-to-video network to be similar to the image-to-image network, so good hyperparameter sets that are already studied in section 4.1.1 will be used directly (here the 3rd set from table1 is used). Based on this, I further study different hyperparameters. First, there are two ways to feed music into the network. One way is to directly use raw data. The raw data is chopped by a window of a certain length and each chopped frame is fed into the network. The other way is to use Mel Frequency Cepstral Coefficient(MFCC) [62] to extract feature vectors from each music window of 40 ms and feed the features to the network. For the first way, impact of different window size is also tested. Second, the impact of cluster numbers n_x and n_y is tested. Third, I test the effect of different number of memory blocks. That is, I either use only LSTM-Conv blocks in the decoders (as in figure5) or use one LSTM-Conv block in the end and normal convolutional layers for the rest.

Metric used for evaluation, besides e_1, e_2 and e_3 as in section 4.1.1, is a video consistence error e_4. Given a generated video sequence $y = y_1, y_2..., y_n$, e_4 is calculated: $e_4 = \sum_{i=1}^{n-1} ||y_i - y_{i+1}||_2$. e_4 is expected to be moderate (neither too large nor too small). A larger value implies that the video is less consistent, while a smaller value implies that the video is more static. However, the optimal value is not known. The result is shown in table2.

Set index	Window size (ms)	Cluster number	Decoder type	e_1	e_2	e_3	e_4
1	40(MFCC)	10	1 LSTM	1.10	0.319	**0.032**	63.0
2	40	10	1 LSTM	1.51	0.340	0.042	34.5
3	80	10	1 LSTM	2.13	0.324	0.045	31.8
4	40	50	1 LSTM	0.83	0.314	0.211	12.2
5	40	50	full LSTM	0.88	0.319	0.281	1.7
6	80	50	full LSTM	0.82	**0.290**	0.346	16.9
7	160	50	1 LSTM	0.88	0.329	0.200	55.5
8	80	100	1 LSTM	0.57	0.329	0.436	13.1
9	80	100	full LSTM	**0.51**	0.309	0.637	3.1

Table 2: quantitative results with different sets of hyperparameters. e_1: error for balanced cluster sizes to be minimized. e_2: error for input variety to be minimized. e_3: error for output diversity to be minimized. e_4: error for video consistence

From table 2, comparing sets 4 with 5, 8 with 9, we can see that full LSTM-Conv blocks yield more consistent videos while only 1 LSTM-Conv block yields more diverge videos. Comparing sets 1 with 2 and 3, we can see that MFCC is generally superior than raw data in terms of e_1,e_2 and e_3 and also yields more diverse videos. Comparing sets 2 with 3, 4 with 7, we see that different sample rates only have a minor impact on the results. Compares sets 2 with 4, 6 with 9, 3 with 8, we see that having more clusters usually means smaller e_1 (more balanced cluster size) but worse e_3 (less diversity in cluster). Having more clusters usually also means more static output, with one exception: set 7, which yields the most diverse output for raw music input. It is also worth mentioning that the clustered input variety e_2 is more consistent through different hyperparameter sets (The worst set is only 15% worse than the best set on e_2, while for e_1 and e_3 they are 76% and 95% respectively).

Qualitative Results

It is important to notice that table 2 cannot capture full characteristics of different sets. Qualitative evaluation is a more direct method. Based on the above results, sets 1, 2, 5 and 7 are compared qualitatively with how much the generated videos match with the input music. Here for the mapping function, only topology mapping is used.

Figure 12 shows the outputs together with the input. A full music piece is fed into the network but here only a small part of 0.8 second in the middle is shown together with the outputs. All hyperparameter sets are unable to generate video that is reasonably similar to the training set but they are worth evaluating as visualization of music nonetheless. (Notice the following analysis may not be so clear with figure 12. The reader might want to watch the full music visualization piece at https://vimeo.com/368386488.) Set 1 with MFCC data type, although performing in general better than sets with raw data in e_1, e_2 and e_3, generates a video that is far too inconsistent. It is also difficult to find a pattern that correspond the music wave. Set 2 generates a green chunk in each image. This green chunk seems to expand a little each time the amplitude of the music gets higher. Set 5 also seems to perform a similar behaviour. But as indicated by a e_4 of only 1.7, the frames bare change much. The correspondence is most clear in Set 7. Focus on the blue part at the center-left of the frames (attached to the bigger green chunk), we see that it shrinks each time antecedent to the growth of the music amplitude. Then it expands after the peak of music wave passes.

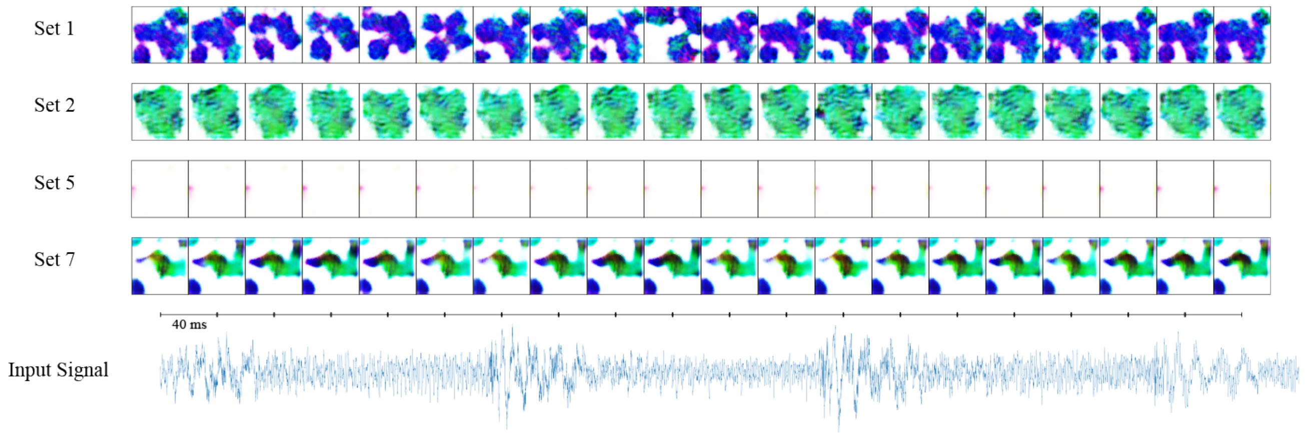

Figure 12: Comparison between outputs from different set. The bottom raw is the sound wave of the music input. Each image in a row represents a frame with 40 ms interval.

By comparing these 4 sets, it is realized that this model is difficult to tune for a good video output in the sense that viewers can see patterns in the video that correspond with the music. This does not necessarily mean the model sometimes cannot find patterns in music, but means the patterns in output videos may be too subtle for the viewers to realize. How to define a good criteria for changes in videos to be not too subtle nor too dramatic so as to ease the hyperparameter tuning process remains a open question.

4.2 Creativity Evaluation

After having some knowledge of the performance of the model, the next important question is: how creative is this model? Using the criteria of section 2.1: (1) Features are distributed and representative; (2) The connections are flexible under different circumstances; (3) Individual differences (flexible and persistent) can be modelled, I evaluate the creative behaviour of the model. The first criteria can be proved with the characteristic of autoencoders: the encoded features are distributed and fully represent the input. Two tasks are designed based on the second and the third criteria respectively.

The first task focuses on the flexibility of the mappings, specifically, the model's behaviour with environmental changes and mental changes. Imagine a scenario where a fashion designer has her own tastes based on many years of experience. Recently she watched a fashion show. Although she did not realize her style was changed a little bit by this fashion show. Then she got to know a new friend who is a famous fashion designer. She was impressed by his talent and wanted to mimic his style. Next, she found her style had been too similar to other designers and decided to do something completely different from the others (particularly, the fashion show and her designer friend) and have her own unique style. However, her old experiences and style are rooted and not very easy to change.

Can the model mimic such human-like behaviours? To make this scenario compatible with our artificial designer, it is first assumed that the designer's job is, when providing a bottom clothing (image), she needs to create a top clothing (image). Her many-year experience can be represented by many pairs of tops and bottoms that she has seen (Guide set 1). The fashion show can also be represented by a set of pairs (Guide set 2). This new set likely has many conflicts with Guide set 1 but also brings some brand new pairs. The friend's style is then defined as Guide set 3. Bottoms from all guide sets are randomly selected. The model first learns from Guide set 1. Since the set is unlikely to cover a person's complete knowledge about clothing, never-referred-to clusters are mapped using topology mapping with a small weight. To simplify the problem, it is also assumed that in each one of the three sets, each image belongs to a unique cluster. The loss to be minimized is:

$$\mathcal{L}^1_{map} = \mathcal{L}^1_{pair} + w_{topo}\mathcal{L}_{topo}$$

where $\mathcal{L}^1_{pair}$ is the loss for matching Guide set 1 and $w_{topo} = 0.1$. All weights in $\mathcal{L}_{pair}$ is a random number in $[0.9, 1.1]$. Creation 1 is generated by minimizing $\mathcal{L}^1_{map}$. The loss after learning Guide set 2 is:

$$\mathcal{L}^2_{map} = \mathcal{L}^1_{pair} + \mathcal{L}^2_{pair} + w_{topo}\mathcal{L}_{topo}$$

where all weights in $\mathcal{L}^2_{pair}$ is a random number in $[0.8, 1.0]$. Creation 2 is generated by minimizing $\mathcal{L}^2_{map}$. After learning Guide set 3:

$$\mathcal{L}^3_{map} = \mathcal{L}^1_{pair} + \mathcal{L}^2_{pair} + \mathcal{L}^3_{pair} + w_{topo}\mathcal{L}_{topo}$$

where all weights in $\mathcal{L}^3_{pair}$ is a random number in $[2, 3]$. Creation 3 is generated by minimizing $\mathcal{L}^3_{map}$. In the end, Guide set 2 and Guide set 3 are treated as negative guides:

$$\mathcal{L}^4_{map} = \mathcal{L}^1_{pair} + \mathcal{L}^2_{pair} + \mathcal{L}^3_{pair} + w_{topo}\mathcal{L}_{topo}$$

where all weights in $\mathcal{L}^2_{pair}$ is a random number in $[-0.2, 0.1]$. Because Guide set 2 is learned in a more subconscious way, it is also hard to discard it. All weights in $\mathcal{L}^3_{pair}$ is a random number in $[-3, -1]$ because this set can be discarded purposefully. Creation 4 is generated by minimizing $\mathcal{L}^4_{map}$.

The result is shown in figure 13. Each pair in Guide set 1 is a top image in the second row and a bottom image of the first row in the same column. Creation 1 is the generations of the model after learning Guide set 1, given the bottoms from the same columns. It also includes the generation from a few never-before-seen bottoms. The same occurs for Guide set 2, Creation 2, Guide set 3, Creation 3. Creation 4 is after the discard of Guide set 2 and 3. We can see that Creation 1 matches well with the designer's experience(Guide set 1). The involvement of Guide set 2, as expected, only directly changes a few generations (e.g. columns 5 and 8). However, this small impact may also change future creations. For example, in row 10, where Guide set 2 contains a pink-red top, after learning Guide set 3 which has no impact on column 10, the artificial designer creates a red top. Creation 3 matches with Guide set 3 well. It is interesting to see that Creation 4 is similar but not identical to Creation 1. This implies that although the artificial designer does not consciously change the influence of its old experience (Guide set 1), its style is shifted slightly during the whole process. This experiment shows not only that the model is able to create flexible mapping functions under different circumstances, but also the fact that behaviours yielded from such flexibility are, in the author's opinion, comparable to human behaviours.

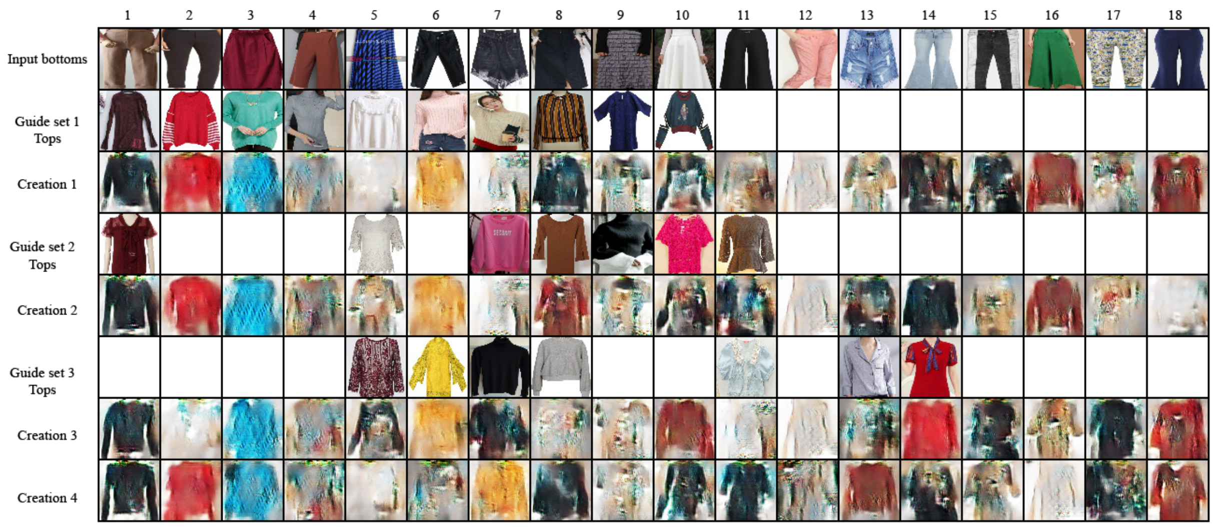

Figure 13: guide sets and creations for flexibility evaluation. Each image of a top of a guide set is paired with the bottom at the same column. Each created image is based on the input bottom at the same column.

The second task evaluates the modelling of persistent and flexible individuals. While a persistent individual tends to find a single best solution, a flexible individual tends to explore for more possibilities, but maybe not as good. This difference can be modelled by the design choices of an evolutionary algorithm(EA). The phenotype is a vector a of length n_x and the oth element $a^o = p$ means the oth cluster of c_x is mapped to the pth cluster of c_y. A persistent individual is modelled by a special EA where the offsprings of a parent a are all vectors that have exactly two elements different from a. The selection process chooses a single best individual from the parent and offsprings. A flexible individual is modelled by another EA where 30 offsprings are generated by mutation of 15 parents. The next generation is selected to be the best 15 offsprings. Thus the parents are discarded. To show the differences between these two type of individuals, the experiment is designed as follow: A first persistent individual learns a mapping function from topology mapping and passes his experience to a second individual. This experience is in the form of paired clusters (a better choice should be paired image frame and music frame, but the cycle-consistence loss L^x_{cyc} of the current state of CDMN is too large to do this). The second individual learns from both topology mapping and the experience. A third persistent individual then learns from topology mapping and the experience from the second individual. The same process is carried out for three flexible individuals.

The generated visual frames from the same music sample from six individuals are shown in figure 14. We can see that persistent individuals 2 and 3 cannot create something new beyond persistent individual 1. While flexible individuals 2 and 3 are not restricted by experiences. However, judged by L_{topo}, later generations of flexible individuals does not necessarily improve over earlier generations. (319, 324 and 311 for individual 1,2 and 3 respectively). Even though persistent individuals have a lower L_{topo} (272 for all individuals), visually it is hard to say if the persistent individuals find "better" mapping functions than the flexible individuals. (A video of these individuals can be found at https://vimeo.com/368390854). In fact, it is hard to say that an elaborated mapping function is better than a completely random mapping function!

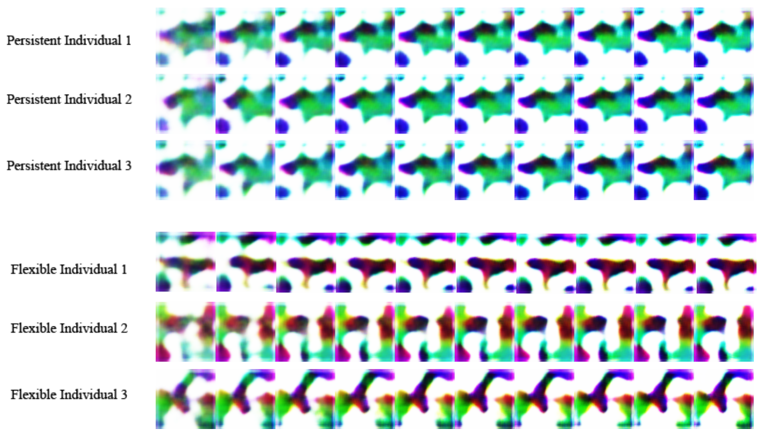

Figure 14: Six video generations from the same music sample. Each row consists of 10 successive frames.

Such difficulty implies that, although the model is able to observe differences in music frames, it fails to quantify such differences and carry quantified differences to the output of the model. For example, judging by a human, a music frame x_1 is believed to be different from both x_2 and x_3 and has more differences from x_3 than from x_2. The mappings found by the model are $x_1 \to y_1$, $x_2 \to y_2$ and $x_3 \to y_3$. A human is barely able to tell that y_1 is closer to y_2 than to y_3. There could be three reasons. First, although figure 8 shows that smaller distances between clusters imply similarities between them in a fairly simple case, with a more complex dataset the model may not be able to find such similarities. Second, the memory blocks may neutralize the similarities between clusters. Since the cluster indices of music frames usually follow a pattern of $[1, 1, 1, 2, 1, 1, 1, 3, 1, 1, 1]$ with each number implies a cluster, the changes to 2 and 3 are just to short and a lag of the memory blocks tend to ignore such changes. Third and the most profound, the nature of discretization and topology mapping make it hard to find a "good" mapping. Discretization makes an originally continuous space "either black or white", forces the grey space to fade out and thus the generations would be biased. While the one-dimensional topology mapping only takes Euclidean distance between clusters into account, meaning multi-dimensional relations between clusters are simplified (Does it make sense that a cluster of red jacks is closer to red T-shirt than to blue jacks?) However, at this point, discretization with topology mapping is the only method for mappings. Future work may want to improve from here.

5 Conclusion

This paper proposed a method for adaptive mappings and generations between domains: CDMN. CDMN encodes discretized cluster features and continuous individual features from one domain, maps these cluster features to cluster features of a second domain and finally generates (decodes) instances in the second domain from these features. Different from previous work, the separation between encoding-decoding functions and mapping functions is closer to human's behaviour and enables the mapping functions to adapt to different situations. The use of mapping criteria based on topological distance and prior pairs helps CDMN to show some complex human-like behaviours.

This paper made an attempt to bridging creative cognition theory and machine learning applications. On one hand, while the literature of Generative Adversarial Networks(GAN) hardly refers itself as creative cognition models, by identifying the process of feature encoding and connecting, this paper locates GAN not only as a statistical model but also a cognitive model. As a GAN application, compared to previous work, this paper achieved a higher level of creativity in terms of better adaptivity and individuality. This implies a bright potential of bringing computers to the next level of creativity by embodying creative cognition theories. On the other hand, as a realization of computational creative theories, compared to previous work, this paper provides a highly automatic method with which the unary action control model with feature distribution and connection is shown to be applicable. It shows the possibilities of using machine learning tools as a convenient and powerful method to build creative models and evaluate creative theories.

Possible future work is suggested in two directions. In the direction of applications, the limitation brought by discretization and topology mapping should be addressed. Although mathematically speaking, these methods are needed to enable the search for mapping functions, they limit the dimensionality of possible mapping functions. Besides naive methods like enlarging cluster numbers, smoothing the transition between clusters or multi-dimensional topology mappings, it is also possible to construct continuous feature space and design mapping functions conditional on feature space coordinates (instead of points in space). It is also not trivial to tune hyperparameters as more in-detail analysis can be performed with higher quality models. In the direction of theories, mapping rules used in this paper can arguably model somewhat but limited human-like behaviors. This is because the mapping rules proposed in this paper are ad-hoc, which is due to the fact that how mapping functions are controlled is not well-studied in cognitive psychology[4]. This paper addresses the importance of such studies to achieve closer-to-human level creativity.

References

[1] Robert J Sternberg and Janet E Davidson. Insight in the gifted. *Educational Psychologist*, 18(1):51–57, 1983.

[2] Elizabeth S Spelke and Katherine D Kinzler. Core knowledge. *Developmental science*, 10(1):89–96, 2007.

[3] Vera Mekern, Bernhard Hommel, and Zsuzsika Sjoerds. Computational models of creativity: a review of single-process and multi-process recent approaches to demystify creative cognition. *Current Opinion in Behavioral Sciences*, 27:47–54, 2019.

[4] Bernhard Hommel and Reinout W Wiers. Towards a unitary approach to human action control. *Trends in cognitive sciences*, 21(12):940–949, 2017.

[5] Agnese Augello, Ignazio Infantino, Antonio Lieto, Giovanni Pilato, Riccardo Rizzo, and Filippo Vella. Artwork creation by a cognitive architecture integrating computational creativity and dual process approaches. *arXiv preprint arXiv:1601.00669*, 2016.

[6] Ana-Maria Olteţeanu and Zoe Falomir. Object replacement and object composition in a creative cognitive system. towards a computational solver of the alternative uses test. *Cognitive Systems Research*, 39:15–32, 2016.

[7] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[8] Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, Colorado Univ at Boulder Dept of Computer Science, 1986.

[9] David DeMers and Garrison W Cottrell. Non-linear dimensionality reduction. In *Advances in neural information processing systems*, pages 580–587, 1993.

[10] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[11] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 172–189, 2018.

[12] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems*, pages 700–708, 2017.

[13] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *Advances in Neural Information Processing Systems*, pages 465–476, 2017.

[14] Joy P Guilford. Creativity: Yesterday, today and tomorrow. *The Journal of Creative Behavior*, 1(1):3–14, 1967.

[15] Liane Gabora. Revenge of the “neurds”: Characterizing creative thought in terms of the structure and dynamics of memory. *Creativity Research Journal*, 22(1):1–13, 2010.

[16] Colin Martindale. Biological bases of creativity. *Handbook of creativity*, 2:137–152, 1999.

[17] Paul T Sowden, Andrew Pringle, and Liane Gabora. The shifting sands of creative thinking: Connections to dual-process theory. *Thinking & Reasoning*, 21(1):40–60, 2015.

[18] Yoed N Kenett, Orr Levy, Dror Y Kenett, H Eugene Stanley, Miriam Faust, and Shlomo Havlin. Flexibility of thought in high creative individuals represented by percolation analysis. *Proceedings of the National Academy of Sciences*, 115(5):867–872, 2018.

[19] Sarnoff Mednick. The associative basis of the creative process. *Psychological review*, 69(3):220, 1962.

[20] Ana-Maria Olteţeanu and Zoe Falomir. comrat-c: a computational compound remote associates test solver based on language data and its comparison to human performance. *Pattern Recognition Letters*, 67:81–90, 2015.

[21] Ana-Maria Olteteanu. *A cognitive systems framework for creative problem solving*. PhD thesis, Universität Bremen, 2016.

[22] Ana-Maria Olteţeanu, Holger Schultheis, and Jonathan B Dyer. Computationally constructing a repository of compound remote associates test items in american english with comrat-g. *Behavior research methods*, 50(5):1971–1980, 2018.

[23] Ivana Kajić, Jan Gosmann, Terrence C Stewart, Thomas Wennekers, and Chris Eliasmith. A spiking neuron model of word associations for the remote associates test. *Frontiers in psychology*, 8:99, 2017.

[24] James L McClelland, David E Rumelhart, PDP Research Group, et al. *Parallel distributed processing*, volume 2. MIT press Cambridge, MA:, 1987.

[25] Timothy T Rogers and James L McClelland. *Semantic cognition: A parallel distributed processing approach*. MIT press, 2004.

[26] Joscha Bach. The micropsi agent architecture. In *Proceedings of ICCM-5, international conference on cognitive modeling, Bamberg, Germany*, pages 15–20. Citeseer, 2003.

[27] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[28] Jun Deng, Zixing Zhang, Erik Marchi, and Björn Schuller. Sparse autoencoder-based feature transfer learning for speech emotion recognition. In *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*, pages 511–516. IEEE, 2013.

[29] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 833–840. Omnipress, 2011.

[30] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *International conference on machine learning*, pages 843–852, 2015.

[31] Margaret A Boden. Creativity in a nutshell. *Think*, 5(15):83–96, 2007.

[32] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[33] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[34] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

[35] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[36] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European Conference on Computer Vision*, pages 702–716. Springer, 2016.

[37] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*, 2016.

[38] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision*, pages 597–613. Springer, 2016.

[39] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *Advances In Neural Information Processing Systems*, pages 613–621, 2016.

[40] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. Midinet: A convolutional generative adversarial network for symbolic-domain music generation. *arXiv preprint arXiv:1703.10847*, 2017.

[41] Donggeun Yoo, Namil Kim, Sunggyun Park, Anthony S Paek, and In So Kweon. Pixel-level domain transfer. In *European Conference on Computer Vision*, pages 517–532. Springer, 2016.

[42] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[43] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[44] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. *arXiv preprint arXiv:1611.02200*, 2016.

[45] Noam Mor, Lior Wolf, Adam Polyak, and Yaniv Taigman. A universal music translation network. *arXiv preprint arXiv:1805.07848*, 2018.

[46] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. *arXiv preprint arXiv:1808.06601*, 2018.

[47] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[48] Lele Chen, Sudhanshu Srivastava, Zhiyao Duan, and Chenliang Xu. Deep cross-modal audio-visual generation. In *Proceedings of the on Thematic Workshops of ACM Multimedia 2017*, pages 349–357. ACM, 2017.

[49] Bin Duan, Wei Wang, Hao Tang, Hugo Latapie, and Yan Yan. Cascade attention guided residue learning gan for cross-modal translation. *arXiv preprint arXiv:1907.01826*, 2019.

[50] Chia-Hung Wan, Shun-Po Chuang, and Hung-Yi Lee. Towards audio to scene image synthesis using generative adversarial network. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 496–500. IEEE, 2019.

[51] Zhiyao Duan, Slim Essid, Cynthia Liem, Gaël Richard, and Gaurav Sharma. Audio-visual analysis of music performances. 2018.

[52] Yang Song, Jingwen Zhu, Xiaolong Wang, and Hairong Qi. Talking face generation by conditional recurrent adversarial network. *arXiv preprint arXiv:1804.04786*, 2018.

[53] Amanda Duarte, Francisco Roldan, Miquel Tubau, Janna Escur, Santiago Pascual, Amaia Salvador, Eva Mohedano, Kevin McGuinness, Jordi Torres, and Xavier Giro-i Nieto. Wav2pix: speech-conditioned face generation using generative adversarial networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 3, 2019.

[54] Ariel Ephrat and Shmuel Peleg. Vid2speech: speech reconstruction from silent video. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5095–5099. IEEE, 2017.

[55] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[56] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pages 802–810, 2015.

[57] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018.

[58] Cloth dataset. `https://tianchi.aliyun.com/competition/entrance/231670/information`.

[59] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[60] Swanhild U Meyer, Michael W Pfaffl, and Susanne E Ulbrich. Normalization strategies for microrna profiling experiments: a 'normal'way to a hidden layer of complexity? *Biotechnology letters*, 32(12):1777–1788, 2010.

[61] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[62] Md Sahidullah and Goutam Saha. Design, analysis and experimental evaluation of block based transformation in mfcc computation for speaker recognition. *Speech communication*, 54(4):543–565, 2012.