スケッチ to イラストな手法、style2paint を読む

コンピュータサイエンス専攻 江畑 拓哉 (201920631)

1 導入

良い感じのお絵描きの補助ツールが欲しい！（友人談）

最近ですと、ClipStudio Paint Pro/Ex が **自動着彩機能** を beta 版で利用可能になるなど、 **お絵描き支援に関する研究** が **市場価値** を生んでいますね（学術的にどうかって？医療データでもない深層学習にお金が下りると思ってんの？）。

そんなわけでスケッチ to イラストみたいな研究論文が海外や国内企業ではそれなりに数が出ているわけなんですが、その中でも即プロダクトに活かせる程度に高度なものとして style2paint というものがあります。これはラフ画からい感じに着彩するというまさに文脈通りの機能を実現するためのもので、実際に対話型のソフトウェアが構築されたものです。これはバージョンが存在しており、2019 年 6 月時点で最新となっているのは v4 です。これに関する論文が、Two-stage Sketch Colorization というものです。

個人的な導入はともかくとして、簡単に論文で述べられている導入の方に踏み込んでみましょう。

> 着彩って難しくて時間かかるし、説得力のある絵を書くには、絵師でも苦労するよね。初心者の人にはお手本となるような、絵師さんには沢山の色の組み合わせを試せるような (例えば髪の色をちょっと変えてみたりとか) 仕組みを作る必要があると思うんだ。
>
> でも着彩にの自動化をしようと思ったら結構多くの課題があって、例えば主題を制限していない状態でのスケッチ画をモデルに理解してもらうのは至難の業だし、スケッチ画は線画に比べれば大分抽象化されているからそこの差を埋めるようにしないといけない、テクスチャとかシェーディングみたいな情報も描かれていないからそれの推測もしなくちゃぁいけない。
>
> ところで最近だと深層学習の技術が発展してきて、Comicolorization なんかの技術が便利な着彩補助の関連研究として出てきているね。でもこいつらはにじみとか色の間違いとかが普通にあったりするんだ。これらを修正できるようにしなきゃいけないんだけど、スケッチ -> 着彩の間に僕たちユーザは上手いこと手を出すことが出来ていなくて、結局自動着彩の技術よりも手打ちの着彩の方がまだまだ人気という感じなんだ。
>
> そんなわけで僕たちは自動着彩じゃなくて半自動着彩のフレームワークを提案するよ。半自動というのはどういうことかっていうと、ユーザが着彩結果をちょいちょい弄って修正できるようにしているっていう意味、つまりスケッチ->着彩 (1)->ユーザの修正->着彩 (2)->. . . ->着彩 (マスター) みたいなことをしたいってことだね。
> 提案したフレームワークは、2 段階に別れた CNN ベースのもので、一つ目は簡単に色をちらして色の概略図みたいなものを作るもの、もう一つはそこから間違いや変な質感になっているものなんかを修正して細かいところを詰めていくものになっているよ。繰り返しになるかもしれないけど、つまりスケッチ -> 着彩という問題をより単純な 2 つのタスクに分離したってことだね。これによってモデルの学習が効率化して、そのモデルが堅牢なものになったことがわかったよ。更に言うと、既存の着彩補助に関する手法で得られた着彩画を修正することもできることがわかったんだ。
>
> それで現実世界でうまく適用できるか試すために GUI アプリを作ってユーザに使ってもらったんだけど、この手法は 2 段階に分かれているとはいえ、ユーザが途中結果を見ながら修正ができることなんかも相まって、ユーザからの満足度は高かったよ。

評価として用いた手法が、実アプリケーションを用いたユーザアンケートだよ。これによって既存手法より良いということを示しています。

最後に簡単に僕達が出来たことをまとめるとこんな感じになるよ。

1. 実世界で作られているスケッチから高品質な着彩ができる 2 段階の着彩モデルを作ったよ
2. 僕たちの提案した手法は、既存の手法と比較して着彩の誤り修正が容易でより高品質な着彩結果を得ることができるようになったよ
3. 実際にユーザに使ってもらえるように、実アプリケーションを作ってみたよ

ほぼほぼ全意訳ですが、導入でおおよそ重要なことを述べきっている論文なのでもりもり書いています。

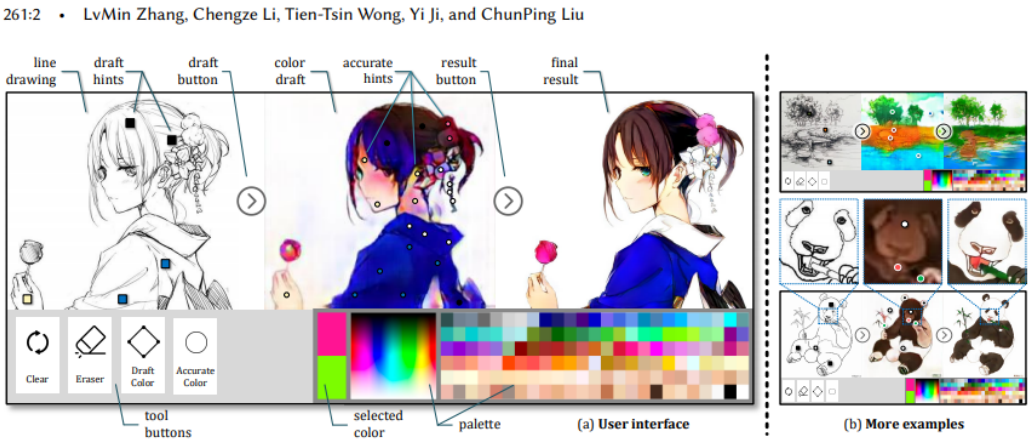

Fig. 2. **The interface of our interactive software.** Users can achieve high-quality results step-by-step through our software. They firstly make a color draft with the draft color tool. After that, they put some more hint points on the generated draft canvas to control details and achieve the final result. © moyan (a), kumantang (all in b).

Figure 1: style2paint 実アプリケーション図

2 関連研究

沢山ありましたので、公開されているものか、読んでみて面白そうだったものか、実際に試して面白かったものだけを抜粋して簡潔に紹介します。

2.1 色伝搬

スケッチ->単色で塗りつぶし

- Lazy Brush

スケッチからいい感じに領域選択して色を塗りつぶす研究。深層学習とかの手法でもないし、恐らく機械学習というわけでもないです。実アプリケーションとしてあるらしく、チュートリアルムービーを見ることが出来ますが、めっちゃ早いです。領域選択系になるので、医療データの癌領域特定なんかに活かせそうですね。大学かなんかでやればいいんじゃないっすかね。

2.2 機械学習ベースの自動着彩

スケッチ->カラー画像 or グレースケール画像->カラー画像
Scribbler と Comicolorization はざっくり言うと、encoder-decoder 型の Generator ＋ GANs の Discriminator のような形を取っています。この部分の研究だと、敵対性ネットワークを用いるのがデフォルトという感じになるようです。

- Scribbler

  スケッチ から カラー画像 or 線画+補助色 から カラー画像を作る手法です。要所要所に色の情報を付け加えることで、より良い画像が出来る、というもので、本手法にはそれなりに近いのかな、と思いました。但しスケッチのクオリティが高すぎるので、初心者のお絵描き補助という目的に沿うことができるようには思えない感じです。

Figure 1. A user can sketch and scribble colors to control deep image synthesis. On the left is an image generated from a hand drawn sketch. On the right several objects have been deleted from the sketch, a vase has been added, and the color of various scene elements has been constrained by sparse color strokes. For best resolution and additional results, see http://scribbler.eye.gatech.edu

Figure 2: Scribbler より引用

- Outline Colorization through Tandem Adversarial Networks.

  グレースケールの画像 から カラー画像を作るための手法です。色彩予測を行うネットワークと、シェーディングを行うネットワークを組み合わせて画像を作り出すネットワークです。グレースケール画像から色の予測を行い、その色予測と、元のグレースケール画像の陰影情報を組み合わせて画像を作る、というモデル (学習には GANs の Discriminator を使う) で style2paint とは違った 2 段階モデルになっています。

- AutoPainter

  スケッチ から カラー画像を作るための手法です。GANs を用いた自動着彩について研究したいなら一度は読みたい、という感じに読みやすい論文です。(というよりは損失関数の定義がすごくわかりやすい形にな収まっている。) pix2pix とのみ比較しているのでどの程度の性能なのかイマイチ理解が出来ないところがあるが、少なくとも pix2pix に対しては圧勝しています。

  面白かったのでもう少し気になったところを書くと、損失関数に画像の滑らかさを付け足す項を追加している点で、それは以下のような式になります。

  $$L_{tv} = \sqrt{(y_{i+1,j} - y_{i,j})^2 + (y_{i,j+1} - y_{i,j})}$$

  この式は他の画像生成系の論文ではあんまり見ないものだったので (というよりくっきりした画像を作るのが GANs の VAE に対する強みの一つなので、それを潰しているようにも捉えられるということが不思議です)、面白みがあるなぁと思いました。

ちなみに一時期 PaintChainer の論文の盗作なのでは？という議論が上がったりもしていましたが、これは恐らく間違いです。

- PaintsChainer シリーズ

  スケッチ->カラー画像を作るための手法です。PFN の出した つよつよ成果物 を引っさげたシリーズです。名前が、たんぽぽ->かな->さつき、となっている 舐め腐った 特徴的なタイトルのものです。論文がないっぽいんですが、これはどういうこっちゃ…？

- cGAN-based Manga Colorization Using a Single Training Image

  グレースケール漫画 から カラー漫画を作るための手法です。物凄い面白い手法を使っているんですが、簡単な特徴に関する説明は このページ にあります。大量のデータで殴りつける最近のビッグデータでグローバルなジャパニーズドリーム () なものとは違い、とても日本人臭い泥にまみれた手法を使っているので、一度読んでみると面白いと思います。

  ちなみにこの手法を用いて低賃金で鬼のように働かされている日本人の漫画家やアニメータを救おう！みたいな 調査論文 が **海外** で出ているのは、これも日本らしくて大好きです。

2.3 画像のスタイル変換

画像のスタイル変換もスケッチ->カラー画像に使えるので関連研究として取り上げられています。

- Discriminative Region Proposal Adversarial Networks for High-Quality Image-to-Image Translation

  GANs を用いた画像のスタイル変換に関する論文。教師あり学習。例えばセグメンテーション画像 (オブジェクトごとに色分けされた画像…？) と写真のような画像との変換、線画から写真のような画像の変換、あるいはそれらの逆元が出来る、と主張されています。実装は こちら から。DRPAN という GANs の応用みたいなモデルを使っているんですが、僕の低脳では理解できませんでした…

Fig. 9. Example results of our DRPAN compared to Pix2pix [14] on aerial to maps (top) and maps to aerial (bottom) tasks.

Figure 3: 論文より引用

- Adversarial Feature Learning

  教師なし学習。これはスタイル変換という文脈ではなく、双方向 GANs を求める研究であることに注目しました。最近ですと Flow-base のモデルが可逆な潜在表現獲得モデルとして有名ですが、GANs でもそのような試みが行われているという意味で非常に興味深かったです。GANs に関する数式がもりもりしているので、GANs の数式をたくさん見てみたい人なんかも読んでみると楽しいかもしれません。というかこの論文が読めれば GANs マスターってくらいには GANs を理解できると思います。

- Unsupervised Image-to-Image Translation Networks

  教師なし学習。実装は こちら 。ドメインを 2 つ仮定して、それぞれのドメインにおける同義の意味を同じ潜在表現として取り扱うことでスタイル変換を行おうとしています。つまり X_1 のドメインからある画像 a と X_2 のドメインから a と同じシチュエーションなある画像 b について考えたときに、それぞれの潜在表現は同じ z ということになります。Generator や Discriminator はスタイルごとに必要になります。つまり X_1 のスタイルの画像についての Discriminator は、 X_1 から得られる画像か、X_2 から得られた画像の潜在表現から G_1 を通して得られた X_1 のスタイルになった画像を判定するものになります。この論文をチョイスした理由は、自然言語含めスタイル変換全般に使えそうな手法だったからです。あとこれは後に拡張されて、2 つのドメインからマルチドメインになったものが出てきていて、非常に 興味深い論文 だったからです (実装)。こっちの論文を読め（自分への圧力）。

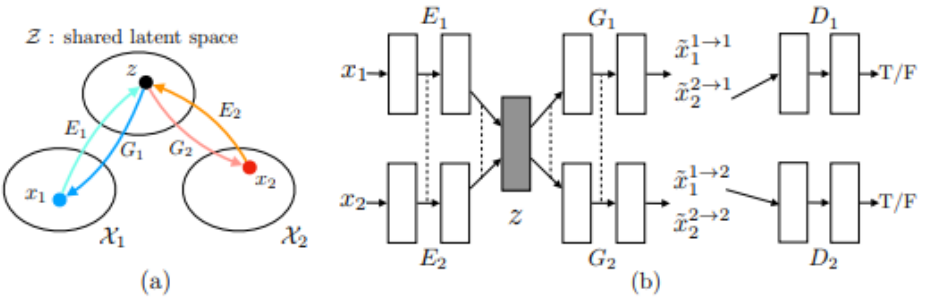

Figure 1: (a) The shared latent space assumption. We assume a pair of corresponding images (x_1, x_2) in two different domains $\mathcal{X}_1$ and $\mathcal{X}_2$ can be mapped to a same latent code z in a shared-latent space $\mathcal{Z}$. E_1 and E_2 are two encoding functions, mapping images to latent codes. G_1 and G_2 are two generation functions, mapping latent codes to images. (b) The proposed UNIT framework. We represent E_1 E_2 G_1 and G_2 using CNNs and implement the shared-latent space assumption using a weight sharing constraint where the connection weights of the last few layers (high-level layers) in E_1 and E_2 are tied (illustrated using dashed lines) and the connection weights of the first few layers (high-level layers) in G_1 and G_2 are tied. Here, $\tilde{x}_1^{1\to1}$ and $\tilde{x}_2^{2\to2}$ are self-reconstructed images, and $\tilde{x}_1^{1\to2}$ and $\tilde{x}_2^{2\to1}$ are domain-translated images. D_1 and D_2 are adversarial discriminators for the respective domains, in charge of evaluating whether the translated images are realistic.

Figure 4: 論文より引用

- CycleGAN

  誰でも知っているので挙げました。解説はこのあたりで見てください。

2.4 画像の色付け

- Let there be Color!

  グレースケール画像 から カラー画像を作るための手法です。早稲田大学の出したグレースケール画像の自動着彩に関する論文。大域・中域・少域特徴を得るためのネットワーク＋色付けのネットワークの 4 つのネットワークをまとめ、彩色画像を作り、それを元のグレースケール画像と組み合わせることでカラー画像を生成します。テレビなんかでも大きく取り上げられたモデルらしいです。大域的・局所的、みたいな文言と最近出てきた OctConv のモデル がなんとなく発想が似ている気がしたのでピックアップしました。

- Real-Time User-Guided Image Colorization with Learned Deep Priors

  グレースケール画像 から カラー画像を作るための手法です。着彩画像に修正が出来ることなど、ほぼほぼ style2paint と同じ仕様になっていますが、こちらは大体の位置に色を置く（塗るではない）することで着彩を行い、スケッチではなくグレースケール画像を入力に用います。かなり良い精度が出ており、これ、 **グリザイユ画法** で使えるんじゃね？と一人思っています。（数年くらい前から日本の一部コミュニティではグリザイユ画法が流行っているという **学術的に価値のない** モチベーションですね）ちなみに GANs のアイデアは使っているのに GANs の損失関数を使わないという面白い内容になっています。GANs を使わないでスタイル変換する論文をこの GANs 時代に提案してくるか…と関心しました。簡単な解説は ここ を読むと良いと思います。そして恐らくこれが最も本論文である style2paint に影響を与えていると思います（具体的には U-net 周りのアーキテクチャがかなり似通っています）。（ ただ見た目の精度が尋常じゃないのに評価手法が PSNR なのが結構気になります ）

  またこの論文では、ユーザの入力に対するシミュレーションも行っており、直接 style2paint のような手法に活かすにはやや大味ではあるものの、この手の手法のデータ収集に関して非常に参考になるものですので、 **一読するべき** でしょう (4 ページの Simulating User Interactions. の部分です)

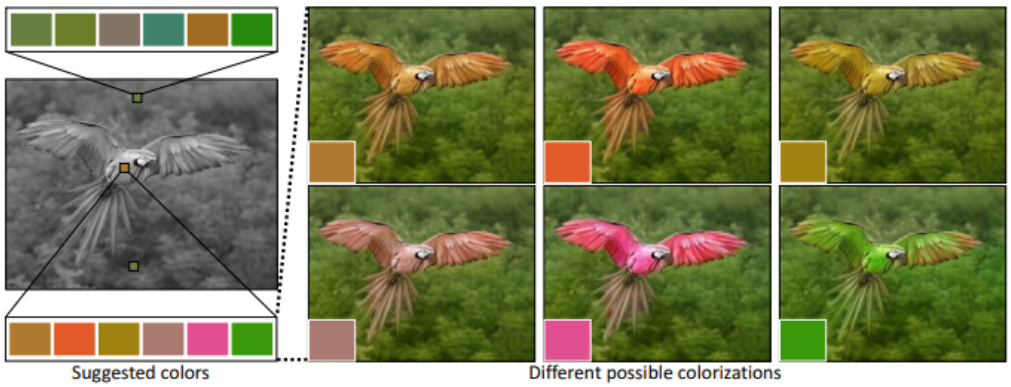

Fig. 3. **Suggested Palette** Our interface provides suggested colors for any pixel, sorted by likelihood, based on the predicted color distribution given by our network. In this example, we show first suggested colors on the background vegetation (top palette), sorted by decreasing likelihood. The suggested colors are common colors for vegetation. We also show the top six suggested colors (bottom palette) of a pixel on the image of the bird. On the right, we show the resulting colorizations, based on the user selecting these top six suggested colors. Photograph of blue-and-yellow macaw by Luc Viatour, 2009.

Figure 5: 論文より引用

3 モデル概要

論文では、提案手法の概要から 2 段階のステップそれぞれの構成、そして訓練データの作成手法についての説明がなされています。これらをざっくりと消化していきます。特に訓練データの作成・獲得手法については **pixiv のサーバダウンを狙ってスクレイピングアタック仕掛けている新進気鋭超頭脳 AI 研究者様** には見ていただきたいものですね。(界隈や大学の印象悪くなるからやめてくれ)

3.1 OverView

2 段階なフレームワークである本手法は、 **drafting stage** と **refinement stage** という名前で 2 つを区別しています。入力のスケッチと最初に与えられるユーザの指示を元に色の構成を決めて、ぱっと色付けをすることが drafting stage での目標になります。そして refinement stage では drafting stage での drafting stage で得られた画像について不正確な色の領域を識別して、追加のユーザからの指示群を元に改良します。これら 2 つの stage に対するモデルは別々に訓練されており、実際に検証を行う際に初めて接続され最終出力までを得ることが出来ます。以下の図 Fig. 3 がフレームワークの全体図です。この 2 段階なフレームワークは複雑な着彩タスクをよりシンプルで目標が明確であるサブタスクに分割したことで、結果的にスケッチと着彩までの距離を狭めます。さらに学習が容易になり、着彩結果の品質が向上します。一方既存の 1 段階な着彩手法では学習が困難であるために、不自然な着彩に対する修正を行うことが出来ません。

訓練に際して **着彩済みなデータセット** として目をつけたものは Danbooru database でした。これに対するスケッチの獲得は、PaintsChainer による線画抽出システムを用いました。またユーザからの入力 (指示) をシミュレートするには、Real-Time User-Guided Image Colorization with LearnedDeep Priors. に用いられている手法を用いました。drafting と refinement 両方で用いられている本質的な手法は、 **GANs** です。Fig. 4 をみると、stacking layer と layer のサイズ、layer 間の接続方式についてわかると思います。訓練時にはおおよそ Adam Optimizer を用いています (where $\beta_1 = 0.9, \beta_2 = 0.99, lr = 1e-5$)。訓練に用いた GPU は Tesla P100 で、バッチサイズは 16 でした (バッチサイズを上げると学習率を下げずに訓練がうまく行く、という論文を google が出していたはずなので、より強い GPU 使って上げてみたいですね。) トレーニングのサンプルデータは、元画像から 224×224 のサイズのパッチにトリミングされます。とはいえ提案手法のモデルは Fully Convolutional Network で構成されているので、本フレームワークの検証段階では **任意の入力サイズをサポートできる** ようになっています。

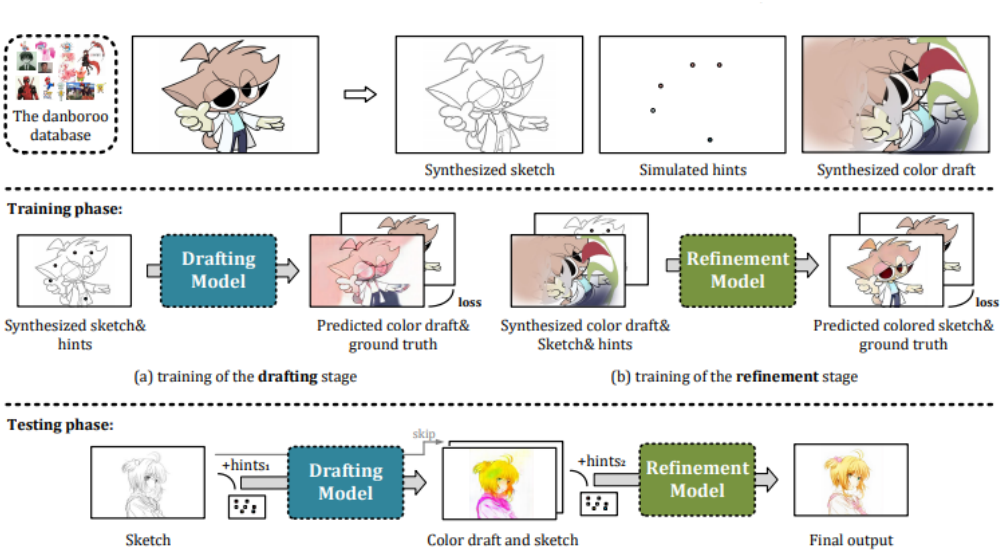

Fig. 3. **Overview of our framework,** including data preparation, the training and testing pipeline. © Senor-G (mouse), style2paints (girl).

Figure 6: Fig.3 論文より引用

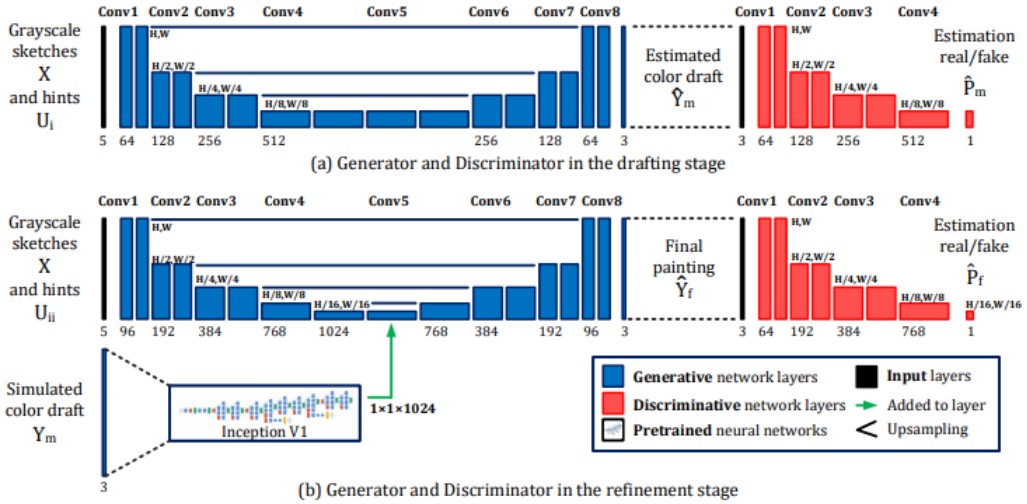

Fig. 4. **Network architectures** of all mentioned models. An $1 \times 1 \times 1024$ vector is extracted from ImageNet Inception [Szegedy et al. 2015] after its first global average pooling. All scale transformations are implemented by subsampling and upsampling. The *add* operation is done by matrix broadcasting. All weights are trainable. Rectified Linear Units [Nair and Hinton 2010] are used as activations. We do not use normalization layers (except for pretrained Inception).

Figure 7: Fig.4 論文より引用

3.2 drafting stage

この stage では入力データであるスケッチから大まかな全体の色構成を決定するという目的で学習されます。高品質な画像を求めているわけではなく、色の多様性を保証できるだけ、ユーザの指示に基づいた色を積極的に散らすことが出来る必要があります。このためにスケッチ x と u_i から大まかな画像 $\hat{y}_m$ を予測するネットワーク network G を提案しています。これの概要は Fig.4 (a) にあります。この大まかながぞうのせいせいについては PaintsChainer など他手法が存在していますが、これらは技術的詳細が明らかにされていません。しかし実験の結果、本手法はそれらと同等以上の性能 (state-of-art な性能) が得られることがわかりました。

スケッチ x とユーザの指示 u_i を入力に、$G(x, u_i)$ で表される FFN (feed-forward network) で 予測画像 $\hat{y}_m$ を出力します。最適化のための目的関数は次の式 (1) になります (概形は **1 ノルム** と **色彩多様性確保のための補正項** 、そして **GANs** ですね)。

$$
\begin{aligned}
arg\min_{G}\max_{D}\mathbb{E}_{x,y_i,y_f\sim P_{data}(x,u_i,y_f)}[\quad & \|y_f - G(x,u_i)\|_1 + \alpha L(G(x,u_i)) \\
& -\lambda log(D(y_f)) - \lambda log(1 - D(G(x,u_i)))] \qquad (1)
\end{aligned}
$$

$$
\begin{aligned}
where & \\
L(x) &= -\Sigma_{c=1}^{3}\frac{1}{m}\Sigma_{i=1}^{m}(x_{c,i} - \frac{1}{m}\Sigma_{i=1}^{m}x_{c,i})^2 \qquad (2) \\
x_{c,i} &= the\ i-th\ element\ on\ the\ c-th\ cannel \\
m &= image\ width\ \times\ height
\end{aligned}
$$

損失関数 L では生成される **色彩の RGB 空間における分散を高める効果** を担っており、これによってより **彩度の高い色をもった** 画像が生成できるようになります。

3.3 refinement stage

drafting stage によって得られた画像はまだ色間違いや不自然な部分 (英語でこれは artifact と言われます) があるため、実用的ではありません。これを修正するために、修正箇所の領域を特定し、それを修正します。このために本フレームワークではユーザから修正箇所の指摘を受けるという仕組みを取っており、その意図を汲み取り制御することが必要になります。これを他制するために、問題点のある色領域を特定・修正するための別の深層学習モデルを提案しました。このモデルはユーザの指示を受け取り、それに従って色間違いや不自然な部分が修正されます。

ところがこのような訓練データを作成することは難しいです。選択肢としては神絵師を札束で殴りつけて draft 画像を修正された画像にしてもらうことですが、これは金も時間もやっべえかかります。またそれによってコンテンツの多様性を確保することも難しいでしょう (神絵師を大量に雇えば良いでしょうが以下略)。あるいは drafting stage から画像を大量に生成してそれを用いるという手法が考えられますが、これを行うと、特定の drafting 画像 に対して過剰適合してしまう可能性があり汎化性能を失う可能性があります。また drafting stage の結果を用いるということはせっかく **意図的に 2 つのタスクに分けた** ものをまとめて訓練してしまっていることになることと同義になるので、望ましいものとは思えません。実際に分離したほうがうまく行くことは、本手法の結果を見ればわかります (実際に drafting stage の画像を用いた refinement stage の学習は提案手法に比べ悪い結果が出ています)。

上記の手法の代わりに本手法では、{color draft, refined painting} の画像ペアを用いた **データセットを大量に自動合成するための手法** を用いました。この合成手法によって、refinement stage のモデルの汎化性能を上げるために役立ち、モデルが異なる種類の不自然な部分を修正することが出来るようになりました。この自動合成手法では、まず draft 画像の潜在的な不自然な箇所の特徴について実際に得られた draft 画像を観察することから始まります。結果として、draft 画像の不自然な箇所はおおよそ以下のような特徴を持っていました。

- 色の間違い

  青い太陽とか、緑な人の顔とかその手の色の間違いです。

- 色の染み出し

  塗りつぶしに失敗した感じです。例えば顔の肌色が背景にまで染み出してしまったことなどが挙げられます。

- ぼやけと歪み

  低い彩度で水彩塗りがぼやけてしまっているとか、一部の領域が変なテクスチャがかかってしまっているとかしました。

以降ではこの 3 点に従うような画像を生成するための手法を説明します。

3.3.1 ランダムな領域切り出しと貼り付け

色の間違いをシミュレートするために、カラー画像からランダムに長方形のパッチを切り出します。パッチのサイズは $64 \times 64 \sim 256 \times 256$ の範囲内で、これは一様分布からサンプリングされます。また色の間違いのランダム性と多様性を確保するために、不規則な形状のパッチを得ることができる領域提案法 (region proposal methods) を用います。領域は入力画像のエッジマップに基づいて抽出されます。まず、ガウスぼかしをかけた画像と元の画像の差を求め、結果をクリッピングすることでシャープでクリーンなエッジマップを得ます。次に取得したエッジマップを平均値に基づいたしきい値に従って 2 値化エッジとして再抽出します。最後に不規則な形状のパッチを抽出するための色領域マスクを得るために Trapped-ball segmentation を実装しました。(Trapped-ball Segmentation についての議論)

これらの 2 つの方法を組み合わせて、全部で 10,000 の異なる画像を抽出します。色の誤りをシミュレートするために、これらのパーツをランダムに回転させて絵の上に重ねて貼り付けます。

つまり簡単に言うと、いい感じに学習データからパーツを持ってきて、画像 y に対して適当に貼り付けることで合成画像 y' を得ます。

3.3.2 ランダムな変形

変形を行うことで、ぼやけと歪みをシミュレートします。まず $[0, 0.1^2]$ 以内の正規分布から得られる乱数 θ_{mn} を値に持つ 2×3 行列 $T_\theta(G)$ を生成します。次に Spatial Transform Layer (STL) (解説) を用いて画像を変形します (STL はどっちかっていうと正規化の手法なんですが、これはとてもユニークな発想ですね、多分)。この変形によって、局所パターンをぼかすことが出来るのと同時に、全体的なノイズ付加ができます。この場合のノイズとは恐らく特徴がボケる、という意味で、画像がぼやける、という意味とはニュアンスが違います。

3.3.3 ランダムな色のスプレー

画像の上にランダムな色をスプレーすることで、色の染み出しをシミュレートします。まず画像内からランダムな色を取り出します。次にいくつかのランダムな線形のパスに従ってランダムに決められた幅 $r \sim uniformly\ distribution \in [64, 128]$ でスプレーします。スプレーの形状は、色の染み出しに似た形状のものを選択しています。

3.3.4 モデルの最適化

上記の 3 つの方法を同時に適用することで、draft 画像 y_m を合成します。スケッチ x 、 ユーザの指示 u_{ii} 、 元の画像 y_f に対して以下の目的関数を使い学習を行います ($\lambda = 0.01$)。前の項が GANs のそれで、後の項は 1 ノルム (MAE, mean absolute error) ですね。 y_m に関する Encoder の初期の重みとして、 ImageNet の inception V1 を与えました。

$$arg \min_G \max_D \mathbb{E}_{x,u_{ii},y_m,y_f \sim P_{data}(x,u_{ii},y_m,y_f)}[-\lambda log(D(y_f)) - \lambda log(1 - D(G(x, u_{ii}, y_m))) + \|y_f - G(x, u_{ii}, y_m)\|_1] \quad (3)$$

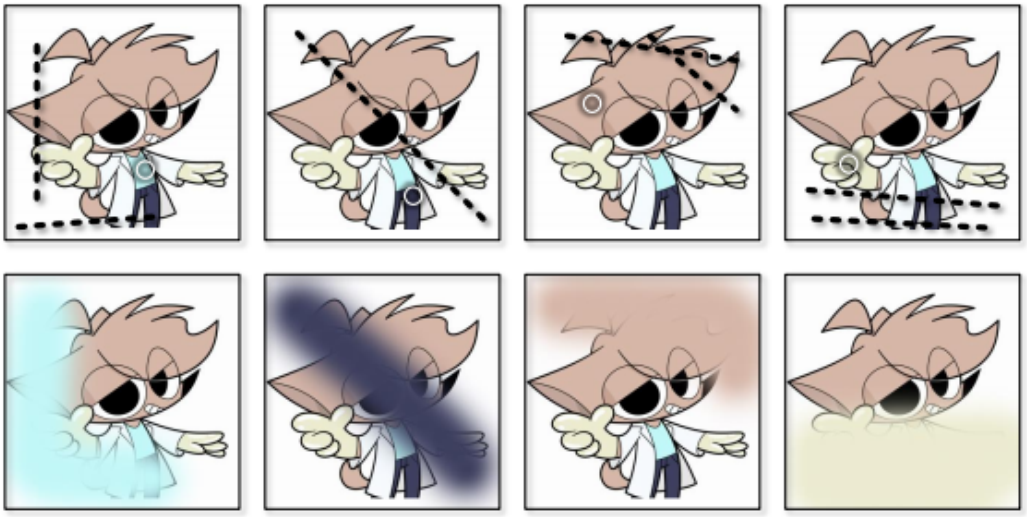

Fig. 7. **Random color spray.** We randomly spray colors to an image to simulate the color bleeding artifact. The sampling points and paths are marked in circles and lines. © Senor-G.

Figure 8: Fig.7 論文より引用

4 評価

5 感想

僕はあんまり画像系の研究はしていないんですが、この論文はとても読みやすい部類であったと思います。最近読んだ TransGaGa の技術を組み合わせるとか Glow や U-Net を参考に Fully Connected Network の構築手法をアップデートする (特に refinement stage の ユーザの指示と画像の組み合わせ部分を TransGaGa の CVAE 項みたいにしてみたら面白そうですね) とか、Refinement Stage を強化学習の分野に持ち込んでみるとか、自動画像生成の部分をもっと詰めてみるとか、色々研究していみたいテーマが見える面白い分野だなぁと思いました。(小並感)

ところでこの論文を読む限り End-to-End 学習には難がありそうなイメージになっているんですが、最近のトレンド的にどうなんでしょう。

最後にこの論文の著者についてです。この著者、物凄いリサーチ力を持っているらしく、先行研究のの実装なんかに積極的に issue を立てて質問に行く (しかも的外れではなくまっとうな質問を！) スタイルはとても尊敬できるなぁと思いました。閉鎖空間で学年やらポストやらで上下関係するスタイルとは違って学問といった雰囲気がしてとても好感が持てます。

追伸・ClipStudio さん とか Pixiv さんとか PFN さんとかで研究してくれないかなぁ（チラッチラッ）