RetroGAN: Translating Unpaired Video Game Images Using CycleGANs

Andrew Rollings, Michael Townsend, Tyler Thurston
Georgia Institute of Technology
arollings3@gatech.edu, mtownsend31@gatech.edu, tthurston7@gatech.edu

Abstract

Video games from the retro era have a signature style due to the 8-bit and 16-bit technology at the time. Limitations on resolution, color, bandwidth, and processing power all created constraints that limited the vision and scope of what video game creators could implement. Using a CycleGAN architecture to allow unpaired translation, we can learn from screenshot data of two separate video game generations (8-bit and 16-bit) and translate images from one generation to the other. Our results show that we can take images from the Nintendo Entertainment System (NES) and Super Nintendo (SNES) and translate screenshots from both consoles into the style of the other. Experiments demonstrate that the CycleGAN architecture can learn many constraints of each system in order to perform relatively accurate translations between generations.

1. Introduction/Background/Motivation

(5 points) What did you try to do? What problem did you try to solve? Articulate your objectives using absolutely no jargon.

In the 1980s, 8-bit consoles were king. These were characterized by limited color palettes and relatively simple 2D graphics. In the 1990s, 16-bit consoles took over and offered increased power, color palettes and the ability to display more complex graphics.

Despite huge advances in technology, retro-games from the 8- and 16-bit eras remain popular, as evidenced by the number of remakes, emulators, modern "Indie" games using pixel art and even specialist magazines dedicating to retro gaming.

Our experiment is based on the observation that the primary difference between 8- and 16-bit games (as exemplified by the NES and SNES, respectively) was in the fidelity of the graphics. Additionally, many of the graphical remasters today view the 16-bit era as the gold standard in pixel-based artwork, given that the graphical capabilities of the SNES still allowed artists to express themselves far more than the limited NES.

Our primary objective is to train two systems, one that can take in an 8-bit NES graphic and "up-convert" it to a realistic 16-bit version and another system that can take in a 16-bit SNES graphic and "down-convert" it to a realistic 8-bit version.

(5 points) How is it done today, and what are the limits of current practice?

Another area of interest is in software emulators that are often used to breathe life into older games. For example, the Nintendo Switch provides a paid emulation service that allows players to access a library of older games. In some cases, the emulators provide graphical filters that allow the appearance of the game to be altered (e.g. a CRT filter, or a resolution upscaling filter). However, currently the only way to improve the graphical color depth of a game is for artists to redraw all of the game's graphics.

(5 points) Who cares? If you are successful, what difference will it make?

If successful, this can allow the graphical content of a particular video game console to be translated into the style of another. Pixel artists are in very high demand and creating high quality pixel art is one of the largest financial strains that can be put on games, and is very time consuming/labor intensive. Pixel artists are often so hard to find that studios will save money by switching to 3D modeling as there are more artists and tools to assist them. This could allow for pixel artists to become more efficient, taking images from a previous game or console generation and translating it to the current project's generation. For instance, if one wished to modernize an 8-bit game, retroGAN enables someone to take the images from the original game and create new baseline screenshots for the sequel. Pixel artists would only need to tweak the translation instead of starting from scratch. There is also the interesting possibility of creating a real-time filter that could do this conversion "on-the-fly" as the game is being played. Additionally, this conversion process is not constrained to translation between NES and SNES, and could be done on other consoles and generations given suitable datasets.

(5 points) What data did you use? Provide details about your data, specifically choose the most important aspects of

your data mentioned here. You don't have to choose all of them, just the most relevant.

Datasets for NES/SNES images did not exist in a suitable form, so we created our own from various screenshot databases on the internet. We implemented our own preprocessing, which included unsupervised color clustering to remove image artifacts, clamping colors to more closely adhere to the console's original color palette, resizing and cropping as appropriate, and other noise reduction techniques. This provided us with a reasonably large set of images that were organized into quality bands based on how much preprocessing was required. We cover this in more detail later in the paper.

2. Approach

(10 points) What did you do exactly? How did you solve the problem? Why did you think it would be successful? Is anything new in your approach?

We attempted to implement translation between video game images from different generations. The primary problem was a lack of paired image data (an image from each generation) since each console generation consists almost solely of different games (with a small number of remakes). We thought CycleGAN had the potential to be a solution as it doesn't require paired data due to the cyclic process for training. That is, a NES image can be converted to a SNES fake by one leg of the cycle, and then the SNES fake can be converted back to a NES "fake fake" by the reverse leg. Then the round-trip images (the NES original and its doubly-converted "fake fake") can be compared for differences. In an ideal system, the real image and the "fake fake" image would be virtually identical. Both directions of the conversion process (NES→SNES and SNES→NES) can be cross-checked in this fashion.

We initially started by taking the driver code[1] for the original CycleGAN paper [7]. We then removed all of the code not directly related to the CycleGAN model and added our own datasets, pre-processing, augmentation, hyperparameter tuning code, and metrics.

Given the almost miraculous results in the literature that have been achieved using CycleGANs with unpaired datasets (and admittedly vast computing resources way beyond our reach), we were reasonably confident that we could use similar techniques to perform a non-trivial conversion between 8-bit and 16-bit screenshots of games. We expected the 16→8-bit conversion to be much simpler than the 8→16-bit conversion, but we were hopeful that we might be able to achieve at least some worthwhile results in the latter case.

Our approach using the existing CycleGAN architecture with our custom dataset is unique and to our knowledge, no one has properly explored this type of translation.

(5 points) What problems did you anticipate? What problems did you encounter? Did the very first thing you tried work?

The main problem we expected was a lack of clean data. While there are plenty of screenshots of games available across the Internet, we could not find any pre-sorted collections of clean images. As such, we ended up writing several custom web scrapers and image processing tools to collect and homogenize screenshots from several game catalog websites. This process is covered in more detail later in the paper.

Additionally, we had concerns that the CycleGAN would be able to capture the "pixel-perfect" constraints and requirements for the conversion process. Previously, CycleGANs have been typically used for experiments that have not required such exact results; for example, the well-known Horse to Zebra results were not conditioned on the exact shades of white and black of the zebra stripes.

Conversely, in our situation, there are several palette restrictions that apply to the NES and SNES consoles. While the main difference between these consoles is their available color palettes, the NES also has several other graphical restrictions that place some difficult to quantify restrictions on the number of colors in a particular areas of the screen. The SNES also has similar restrictions, although nowhere near as severe. This was anticipated to be very difficult to generalize without magnitudes more data that we had available to us (approximately 100,000 images of varying quality between the two consoles). We didn't see great results in our first attempt, however after performing some additional preprocessing (covered later in this paper), we started seeing interesting results with convincing translations very early on.

Important: Mention any code repositories (with citations) or other sources that you used, and specifically what changes you made to them for your project.

2.1. NES / SNES Palette Considerations

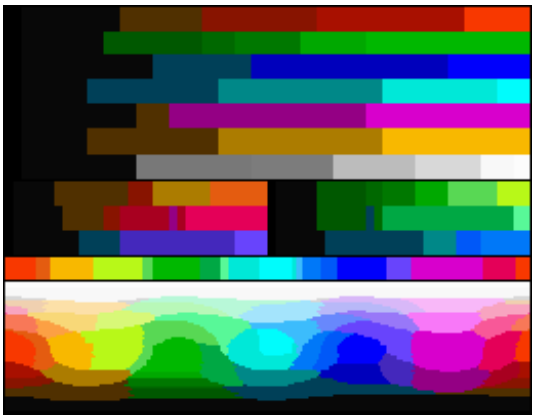
The NES palette comprises 56 fixed colors, whereas the SNES is more flexible with a full 15-bit RGB palette (with 5 bits for each of the red, green and blue channels), as shown in 1a and 1b respectively.

As previously mentioned, there are some additional restrictions to how these colors can be used, but we do not cover them here; they are of more concern to emulator writers, and it was hoped that – given enough data – the CycleGAN would be able to figure these out for itself.

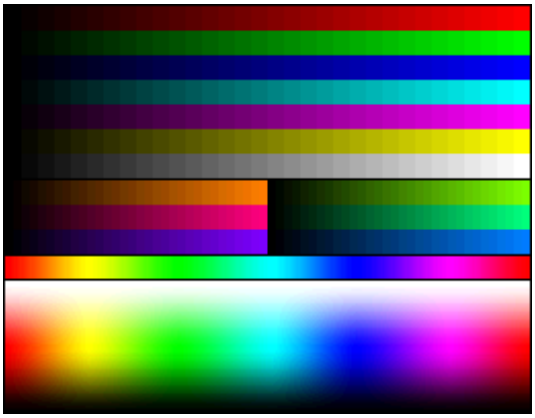
As such, for an effective conversion, the GAN must learn to be conform to each palette. In the case of SNES→NES, this could be as simple as a naive nearest color reduction from the 15-bit SNES palette to the fixed NES palette. Sim-

[1] https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix, Commit Id 00d5574908eb66fe0127b32d7b030001453f21d0

ilarly, for the NES→SNES conversion, the GAN could simply ensure that the chosen colors fit the 15-bit RGB restriction. However, we hoped for (and observed) more complex behaviors than this, implying that the GAN was able to somewhat successfully extract more of the stylistic differences between NES and SNES graphics. This will be discussed in later sections.



(a) NES 56 Color Palette



(b) SNES 15-bit Palette

Figure 1: Color palettes for NES and SNES

2.2. Generative Models and CycleGANs

For our main architecture, we chose the CycleGAN architecture for the many positives it brings. First, being a Generative Model, it brings us the ability to generate new models via sampling from our implicit density estimation. On top of that, we also were drawn by the advantages of specifically Generative Adversarial Networks(GANs)[6]. GANs provide several benefits over other generative models, including not needing labeled data as GANs are unsupervised. They are also flexible on the types of data they ingest. Most importantly, since they sample from their density instead of doing any kind of averaging, GANs result in the sharpest images over other generative models. This is a very important property as retro graphics involve small, sharp pixel placement which makes this a vital quality for this dataset.

Finally, we chose the specific framework of CycleGAN[7], as it provides the ability to translate between two datasets without the need for paired images. Instead of the usual GAN architecture of a single generator and discriminator, the CycleGAN architecture uses mappings: G and F with the goal of having G learn a mapping from domain $X \rightarrow Y$ such that $G(X)$ is indistinguishable from the distribution Y using adversarial loss. However, due to the low constraints of unpaired data, a second, inverse mapping $F : Y \rightarrow X$ is created to induce a cycle consistency loss to push $F(G(X)) \approx X$.

$$G : X \rightarrow Y, F : Y \rightarrow X : F(G(X)) \approx X$$

Figure 2: CycleGAN mappings

These mappings are combined with two *cycle consistency losses* that can capture the intuition that translating from one domain to another and back again should result to something similar to the original input.

Additionally, an *identity loss* is used to provide color/hue stability. This is defined as $F(X) \approx X$ and $G(Y) \approx Y$. The identity loss is combined with the *cycle consistency loss* proportionally as determined by a bias hyperparameter.

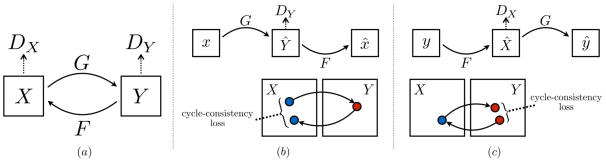


Figure 3: CycleGAN Architecture

The capabilities of GANs combined with the new flexibility of CycleGANs to allow learning from unpaired images, we have the capability to attempt image-to-image translation on our datasets between screenshot data of the NES and SNES.

3. Experiments and Results

(10 points) How did you measure success?

The CycleGAN framework allowed for significant configurability of hyperparameters, and we took full advantage of the flexibility provided. As such, for our first pass success measure, we selected for the hyperparameters that showed the smallest amount of loss for at least *one* direction of the cycle. This gave us a shortlist of candidates to examine that we evaluated manually to determine which to investigate further. One interesting (but not entirely unexpected)

observation we found was that the cycle performance was not even; a set of hyperparameters that performed well for the NES→SNES conversion would not necessarily perform well for the SNES→NES conversion, and vice versa.

For each promising candidate set, we then ran a customized metric to evaluate the final images for their adherence to the hardware restrictions of each console. We produced several iterations of this metric, before finally settling on Algorithm 1. This metric scores an image based on how closely the pixels match the target console palette modulated by the percentage of image pixels that match the target palette exactly. This modulation was necessary to compensate for the fact that the SNES palette has many more entries, and consequently, it was easier for the algorithm to score highly as any selected color was never too far away from a valid palette entry in RGB space.

Note that this was applied as a post-training evaluation metric. An idea for future work would be to implement this metric as part of the loss function; unfortunately, time constraints prevented us from performing this investigation ourselves.

What experiments were used? What were the results, both quantitative and qualitative? Did you succeed? Did you fail? Why? Justify your reasons with arguments supported by evidence and data.

Should probably save this entire section for after we've established our final metrics and have screenshots to show.

Important: This section should be rigorous and thorough. Present detailed information about decision you made, why you made them, and any evidence/experimentation to back them up. This is especially true if you leveraged existing architectures, pre-trained models, and code (i.e. do not just show results of fine-tuning a pre-trained model without any analysis, claims/evidence, and conclusions, as that tends to not make a strong project).

4. Other Sections

You are welcome to introduce additional sections or subsections, if required, to address the following questions in detail.

(5 points) Appropriate use of figures / tables / visualizations. Are the ideas presented with appropriate illustration? Are the results presented clearly; are the important differences illustrated?

This seems like free points, just properly caption tables and figures. We just need to make sure when we're showing before-after images to make sure we show the metric improving and talk about it in analysis.

(5 points) Overall clarity. Is the manuscript self-contained? Can a peer who has also taken Deep Learning understand all of the points addressed above? Is sufficient detail provided?

This seems like free points. We probably just need to briefly tough on GANs and give a proper section on how CycleGANs work, as well as some detail between NES and SNES color palettes.

(5 points) Finally, points will be distributed based on your understanding of how your project relates to Deep Learning. Here are some questions to think about:

This also seems free. We should probably go into the architecture of each GAN just so we can estimate how many parameters we have. (should probably wait until final HPs are done)

What was the structure of your problem? How did the structure of your model reflect the structure of your problem?

Our problem had two main pieces, a set of NES images and a set of SNES images, without a successful way to map from one domain to the other. The CycleGAN model fit our needs perfectly. Since it is composed of two GANs and two discriminators, it learned a mapping between domains.

What parts of your model had learned parameters (e.g., convolution layers) and what parts did not (e.g., post-processing classifier probabilities into decisions)?

Both GANs and discriminators had learned parameters. The model trained one GAN to make NES images and the other to make SNES images while the two discriminators were trained to detect real and fake NES and SNES images, respectively.

What representations of input and output did the neural network expect? How was the data pre/post-processed?

This is Andrew's section. Should talk about the pre-processing we did.

What was the loss function?

TODO: Depends on best one picked via hyperparameter. We might want to mention the losers in passing.

Did the model overfit? How well did the approach generalize?

This is probably a good way to talk about our augmentation and how it helps with over-fitting.

What hyperparameters did the model have? How were they chosen? How did they affect performance? What optimizer was used?

The original CycleGAN architecture did not have a method for finding the best hyperparameters. We built a wrapper around the train function that allowed us to randomize the available hyperparameters and run multiple training iterations, evaluating each one and retaining the best performing.

TODO: Add in the hyperparameters we ended up using.

What Deep Learning framework did you use?

PyTorch

What existing code or models did you start with and what did those starting points provide?

We used a CycleGAN framework, which is composed of two GANs and two discriminators. The first GAN generates an image, which is passed to the first discriminator, making a prediction on whether the image received is real or fake. The image is then passed to a second GAN, which tries to remake the original image. The output is then passed to a second discriminator, which evaluates whether it received a real or fake image.

The framework provided a first pass implementation for us, giving time to align the framework to our specific problem.

Briefly discuss potential future work that the research community could focus on to make improvements in the direction of your project's topic.

This is an interesting one, we could talk about using sprites, more data, other consoles, using the metrics we made with loss, all sorts of stuff.

Additional work could focus on expanding the consoles that the framework could work on. For example, mapping from 16-bit to 32-bit or even mapping larger jumps from 16-bit to 64-bit. There would be added complexity to this, which may result in needing larger training sets or alternate loss functions.

Alternate loss functions in particular are an intersting area for further research. The CycleGAN architecture was built for the general problem on mapping images from one domain to another. However, for the specific problem of mapping images from one console generation to another, loss functions could be more precise. Others could research using the loss function to additionally model the console's graphical constraints, learning the exact color palette of each console generation.

5. Work Division

Please add a section on the delegation of work among team members at the end of the report, in the form of a table and paragraph description. This and references do ***NOT*** *count towards your page limit. An example has been provided in Table 2.*

6. Miscellaneous Information

The rest of the information in this format template has been adapted from CVPR 2020 and provides guidelines on the lower-level specifications regarding the paper's format.

6.1. Language

All manuscripts must be in English.

6.2. Paper length

Papers, excluding the references section, must be no longer than six pages in length. The references section will not be included in the page count, and there is no limit on the length of the references section. For example, a paper of six pages with two pages of references would have a total length of 8 pages.

6.3. The ruler

The LaTeX style defines a printed ruler which should be present in the version submitted for review. The ruler is provided in order that reviewers may comment on particular lines in the paper without circumlocution. If you are preparing a document using a non-LaTeX document preparation system, please arrange for an equivalent ruler to appear on the final output pages. The presence or absence of the ruler should not change the appearance of any other content on the page. The camera ready copy should not contain a ruler. (LaTeX users may uncomment the `\cvprfinalcopy` command in the document preamble.) Reviewers: note that the ruler measurements do not align well with lines in the paper — this turns out to be very difficult to do well when the paper contains many figures and equations, and, when done, looks ugly. Just use fractional references (e.g. this line is 095.5), although in most cases one would expect that the approximate location will be adequate.

6.4. Mathematics

Please number all of your sections and displayed equations. It is important for readers to be able to refer to any particular equation. Just because you didn't refer to it in the text doesn't mean some future reader might not need to refer to it. It is cumbersome to have to use circumlocutions like "the equation second from the top of page 3 column 1". (Note that the ruler will not be present in the final copy, so is not an alternative to equation numbers). All authors will benefit from reading Mermin's description of how to write mathematics: `http://www.pamitc.org/documents/mermin.pdf`.

Finally, you may feel you need to tell the reader that more details can be found elsewhere, and refer them to a technical report. For conference submissions, the paper must stand on its own, and not *require* the reviewer to go

to a techreport for further details. Thus, you may say in the body of the paper "further details may be found in [5]". Then submit the techreport as additional material. Again, you may not assume the reviewers will read this material.

Sometimes your paper is about a problem which you tested using a tool which is widely known to be restricted to a single institution. For example, let's say it's 1969, you have solved a key problem on the Apollo lander, and you believe that the CVPR70 audience would like to hear about your solution. The work is a development of your celebrated 1968 paper entitled "Zero-g frobnication: How being the only people in the world with access to the Apollo lander source code makes us a wow at parties", by Zeus *et al*.

You can handle this paper like any other. Don't write "We show how to improve our previous work [Anonymous, 1968]. This time we tested the algorithm on a lunar lander [name of lander removed for blind review]". That would be silly, and would immediately identify the authors. Instead write the following:

> We describe a system for zero-g frobnication. This system is new because it handles the following cases: A, B. Previous systems [Zeus et al. 1968] didn't handle case B properly. Ours handles it by including a foo term in the bar integral.
>
> ...
>
> The proposed system was integrated with the Apollo lunar lander, and went all the way to the moon, don't you know. It displayed the following behaviours which show how well we solved cases A and B: ...

As you can see, the above text follows standard scientific convention, reads better than the first version, and does not explicitly name you as the authors. A reviewer might think it likely that the new paper was written by Zeus *et al*., but cannot make any decision based on that guess. He or she would have to be sure that no other authors could have been contracted to solve problem B.

FAQ

Q: Are acknowledgements OK?
A: No. Leave them for the final copy.

Q: How do I cite my results reported in open challenges?
A: To conform with the double blind review policy, you can report results of other challenge participants together with your results in your paper. For your results, however, you should not identify yourself and should not mention your participation in the challenge. Instead present your results referring to the method proposed in your paper and draw conclusions based on the experimental comparison to other results.

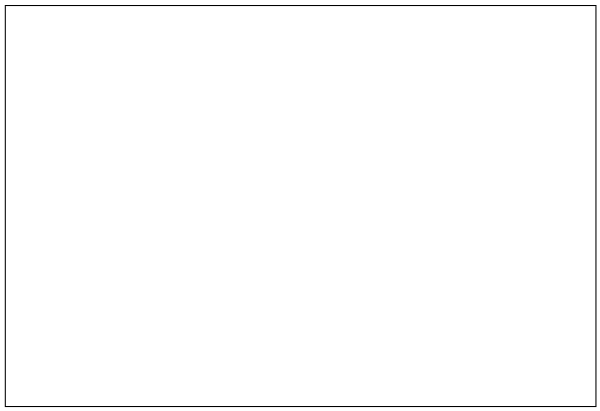


Figure 4: Example of caption. It is set in Roman so that mathematics (always set in Roman: $B \sin A = A \sin B$) may be included without an ugly clash.

6.5. Miscellaneous

Compare the following:
`$conf_a$` $conf_a$
`$\mathit{conf}_a$` conf_a
See The TEXbook, p165.

The space after *e.g.*, meaning "for example", should not be a sentence-ending space. So *e.g.* is correct, *e.g.* is not. The provided `\eg` macro takes care of this.

When citing a multi-author paper, you may save space by using "et alia", shortened to "*et al.*" (not "*et. al.*" as "*et*" is a complete word.) However, use it only when there are three or more authors. Thus, the following is correct: " Frobnication has been trendy lately. It was introduced by Alpher [1], and subsequently developed by Alpher and Fotheringham-Smythe [2], and Alpher *et al.* [3]."

This is incorrect: "... subsequently developed by Alpher *et al.* [2] ..." because reference [2] has just two authors. If you use the `\etal` macro provided, then you need not worry about double periods when used at the end of a sentence as in Alpher *et al.*

For this citation style, keep multiple citations in numerical (not chronological) order, so prefer [2, 1, 4] to [1, 2, 4].

6.6. Formatting your paper

All text must be in a two-column format. The total allowable width of the text area is $6\frac{7}{8}$ inches (17.5 cm) wide by $8\frac{7}{8}$ inches (22.54 cm) high. Columns are to be $3\frac{1}{4}$ inches (8.25 cm) wide, with a $\frac{5}{16}$ inch (0.8 cm) space between them. The main title (on the first page) should begin 1.0 inch (2.54 cm) from the top edge of the page. The second and following pages should begin 1.0 inch (2.54 cm) from the top edge. On all pages, the bottom margin should be 1-1/8 inches (2.86 cm) from the bottom edge of the page for 8.5×11-inch paper; for A4 paper, approximately 1-5/8

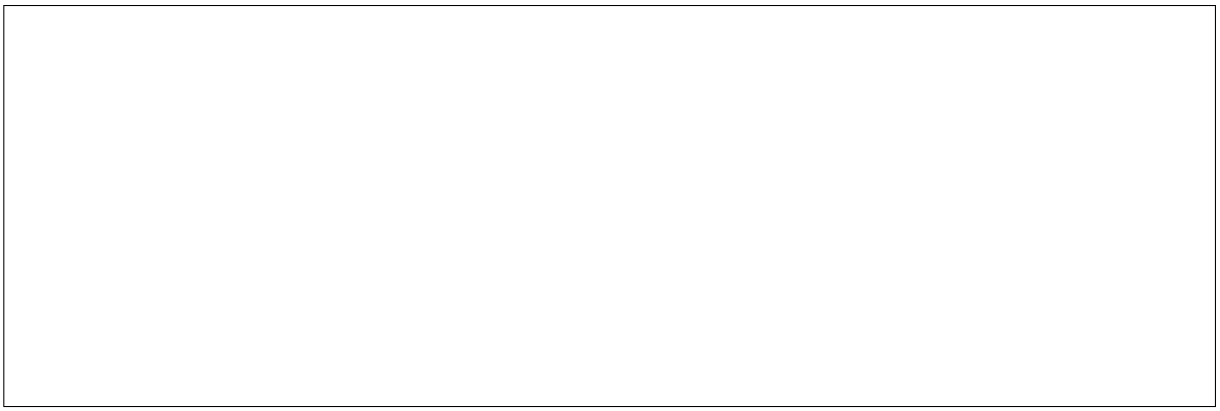


Figure 5: Example of a short caption, which should be centered.

inches (4.13 cm) from the bottom edge of the page.

6.7. Margins and page numbering

All printed material, including text, illustrations, and charts, must be kept within a print area 6-7/8 inches (17.5 cm) wide by 8-7/8 inches (22.54 cm) high.

6.8. Type-style and fonts

Wherever Times is specified, Times Roman may also be used. If neither is available on your word processor, please use the font closest in appearance to Times to which you have access.

MAIN TITLE. Center the title 1-3/8 inches (3.49 cm) from the top edge of the first page. The title should be in Times 14-point, boldface type. Capitalize the first letter of nouns, pronouns, verbs, adjectives, and adverbs; do not capitalize articles, coordinate conjunctions, or prepositions (unless the title begins with such a word). Leave two blank lines after the title.

AUTHOR NAME(s) and AFFILIATION(s) are to be centered beneath the title and printed in Times 12-point, non-boldface type. This information is to be followed by two blank lines.

The ABSTRACT and MAIN TEXT are to be in a two-column format.

MAIN TEXT. Type main text in 10-point Times, single-spaced. Do NOT use double-spacing. All paragraphs should be indented 1 pica (approx. 1/6 inch or 0.422 cm). Make sure your text is fully justified—that is, flush left and flush right. Please do not place any additional blank lines between paragraphs.

Figure and table captions should be 9-point Roman type as in Figures 4 and 5. Short captions should be centred.
Callouts should be 9-point Helvetica, non-boldface type. Initially capitalize only the first word of section titles and first-, second-, and third-order headings.

FIRST-ORDER HEADINGS. (For example, **1. Introduction**) should be Times 12-point boldface, initially capitalized, flush left, with one blank line before, and one blank line after.

SECOND-ORDER HEADINGS. (For example, **1.1. Database elements**) should be Times 11-point boldface, initially capitalized, flush left, with one blank line before, and one after. If you require a third-order heading (we discourage it), use 10-point Times, boldface, initially capitalized, flush left, preceded by one blank line, followed by a period and your text on the same line.

6.9. Footnotes

Please use footnotes[2] sparingly. Indeed, try to avoid footnotes altogether and include necessary peripheral observations in the text (within parentheses, if you prefer, as in this sentence). If you wish to use a footnote, place it at the bottom of the column on the page on which it is referenced. Use Times 8-point type, single-spaced.

6.10. References

List and number all bibliographical references in 9-point Times, single-spaced, at the end of your paper. When referenced in the text, enclose the citation number in square brackets, for example [4]. Where appropriate, include the name(s) of editors of referenced books.

6.11. Illustrations, graphs, and photographs

All graphics should be centered. Please ensure that any point you wish to make is resolvable in a printed copy of the paper. Resize fonts in figures to match the font in the body text, and choose line widths which render effectively in print. Many readers (and reviewers), even of an electronic copy, will choose to print your paper in order to read it.

[2]This is what a footnote looks like. It often distracts the reader from the main flow of the argument.

Method	Frobnability
Theirs	Frumpy
Yours	Frobbly
Ours	Makes one's heart Frob

Table 1: Results. Ours is better.

You cannot insist that they do otherwise, and therefore must not assume that they can zoom in to see tiny details on a graphic.

When placing figures in LaTeX, it's almost always best to use `\includegraphics`, and to specify the figure width as a multiple of the line width as in the example below

```
\usepackage[dvips]{graphicx} ...
\includegraphics[width=0.8\linewidth]
                {myfile.eps}
```

6.12. Color

Please refer to the author guidelines on the CVPR 2020 web page for a discussion of the use of color in your document.

References

[1] FirstName Alpher. Frobnication. *Journal of Foo*, 12(1):234–778, 2002. 6

[2] FirstName Alpher and FirstName Fotheringham-Smythe. Frobnication revisited. *Journal of Foo*, 13(1):234–778, 2003. 6

[3] FirstName Alpher, FirstName Fotheringham-Smythe, and FirstName Gamow. Can a machine frobnicate? *Journal of Foo*, 14(1):234–778, 2004. 6

[4] Authors. The frobnicatable foo filter, 2014. Face and Gesture submission ID 324. Supplied as additional material `fg324.pdf`. 6, 7

[5] Authors. Frobnication tutorial, 2014. Supplied as additional material `tr.pdf`. 6

[6] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. 3

[7] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017. 2, 3

Student Name	Contributed Aspects	Details
Andrew Rollings	Dataset Pre-processing, Architecture Design, Implementation, HP tuning, Report	Came up with initial idea of using CycleGANs and Video Game images. Created dataset and associated pre-processing. Implemented PyTorch dataset loader compatible with CycleGAN Tuned GAN hyper-parameters. Updated metrics calculation and hyper-parameter tuning wrapper code. Helped with report.
Michael Townsend	Implementation, HP tuning, Metrics, Report	Implemented first pass of custom metrics calculations. Tuned GAN hyper-parameters. Designed and implemented code wrapper for hyper-parameter tuning. Helped with report.
Tyler Thurston	Implementation, Metrics, HP tuning, Version Control, Report	Setup team in version control. Git Repository organization and maintenance. Came up with NES color palette metric. Graphing coding. Tuned GAN hyper-parameters. Overall planning and scheduling. Helped with report.

Table 2: Contributions of team members.

$console\_palette \leftarrow$ target console color palette normalized RGB values
$max\_rgb \leftarrow$ corners of RGB space for $all(i, j, k), i = \{0, 1\}, j = \{0, 1\}, k = \{0, 1\}$
$m_bias \leftarrow 1/$number of colors in target console color palette
$total\_score \leftarrow 0$

for each candidate image **do**
  $img\_rgb \leftarrow$ per pixel normalized RGB values
  $min\_dists \leftarrow$ min distance between img_rgb and $console\_palette$
  $max\_dists \leftarrow$ min distance between img_rgb and max_rgb values.
  $match\_pct \leftarrow \sum_{i=1..n}[1 \text{ if } (min\_dists_i < 1) \text{ else } 0]/n$
  $bias\_factor \leftarrow max(0, min(1, m\_bias + (1 - m\_bias) \times match\_pct)$
  $img\_score \leftarrow (\sum_{i=1..n}(max\_dists) - \sum_{i=1..n}(min\_dists))/\sum_{i=1..n}(max\_dists)$
  $total\_score \leftarrow total\_score + img\_score \times bias\_factor$
end for
return $total_score/$number of candidate images

Algorithm 1: Metric for evaluating conversion quality