بسم الله الرحمن الرحیم

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

تمرین درس هوش مصنوعی در سیستم‌های نهفته – مهر ۱۴۰۲

اهداف

آشنایی با روال پیاده‌سازی شبکه‌های عصبی روی پلتفرم‌هایی با منابع محدود است. این روال شامل پیاده‌سازی سطح بالا (کد پایتان[۱])، فشرده‌سازی مدل و استفاده از نتایج آن برای پیاده‌سازی سطح پایین (سیستم وریلاگ[۲]) است.

۱- مقدمه

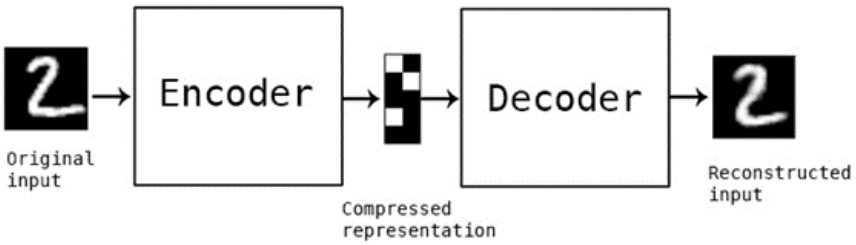
برای دستیابی به هدف تعیین شده، از میان طیف گسترده‌ای از انواع شبکه‌های عصبی موجود، ما یک مثال ساده و کاربردی از شبکه‌ی MLP درنظر گرفته‌ایم. در ادامه توضیح مختصری درباره‌ی این دسته از شبکه‌ها داده می شود و سپس در بخش‌های بعدی نیازمندی‌های لازم برای انجام تمرین ذکر می شود.

۱-۱- شبکه MLP

شبکه عصبی چند لایه پرسپترون[۳] (MLP) یکی از شبکه‌های عصبی مصنوعی چرکاربرد است که در یادگیری ماشین و یادگیری عمیق استفاده می‌شود. شبکه های MLP به خاطر توانایی در مدیریت یک دسته وسیع از وظایف، شامل مسائل دسته‌بندی و رگرسیون، شناخته می‌شوند. به دلیل سبک وزن بودن و محاسبات کم، شبکه‌ها MLP، با استفاده از ماژول‌های MACT[4] به راحتی قابل پیاده‌سازی برروی سخت‌افزار هستند. این مدل‌ها، در کاربردهای زیادی در زمینه‌های بسیار متنوع، از جمله پردازش تصویر و تشخیص لبه[۵] تصویر، جهت حل مسائل راه یافته اند.

۱-۲- رمزگذار خودکار

یکی از کاربردهای مهم شبکه‌های MLP، طراحی رمزگذار خودکار[۶] است. "اتوکدینگ" یک الگوریتم فشرده‌سازی داده است که از یک رمزنگار و یک رمزگشا تشکیل شده است. شکل 1 شمایی از یک اتوانکودر را نشان می‌دهد. این رمزگذار و بازگشایی به‌گونه‌ای عمل می‌کنند که به‌جای طراحی توسط انسان، به صورت ۱) داده گرا[۷] ، ۲) تلفیقی[۸] و ۳) خودکار، عملکرد خود را از مثال‌ها یاد می‌گیرند.

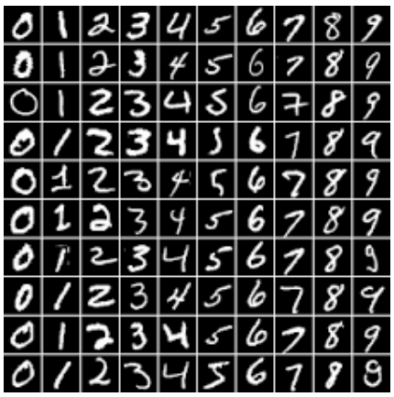


شکل ۱- رمزگذار خودکار

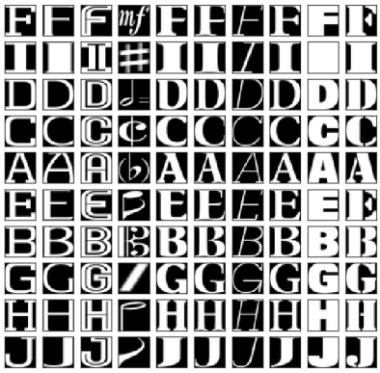
۱ Python
۲ SystemVerilog
۳ Multi-layer Perceptron
۴ Multiply And Accumulator
۵ edge
۶ autoencoder
۷ Data-specific
۸ lossy

۳-۱- دیتاست

در این تمرین ما از یک شبکه عصبی MLP برای طراحی رمزگذار خودکار بر روی مجموعه داده[۹]های MNIST [1] و NotMNIST [2] استفاده می‌کنیم. در شکل2 و شکل 3 به ترتیب نمونه‌هایی از داده‌های MNIST و NotMNIST نشان داده شده است.



شکل۲ - بخشی از مجموعه داده MNIST



شکل ۳- بخشی از مجموعه داده NotMNIST

امروزه دو کاربرد عملی جالب از اتوانکودرها، پاکسازی داده و کاهش ابعاد برای تصویرسازی داده‌ها است. با محدودیت‌های مناسب در ابعاد و انباشتگی، اتوانکودرها می‌توانند نگاشت[۱۰] های داده را یاد بگیرند که جذاب‌تر از روشی مانند PCA [۱۱] یا تکنیک‌های اساسی دیگر هستند [3].

۴-۱- فشرده‌سازی

تاکنون با شبکه‌های MLP و یکی از کاربردهای آن آشنا شدید. اگرچه این مدل‌ها کوچک هستند و به راحتی با دقت بیتی بالا بر روی پلتفرم‌هایی با منابع محدود قابل پیاده‌سازی هستند، ولی در دنیای واقعی شبکه‌های بزرگی وجود دارند که پیاده‌سازی آن‌ها با دقت بیتی بالا، به دلیل حجم بسیار زیاد پارامترهای موجود، که نیازمند تبادلات حافظه‌ای زیاد و محاسبات سنگین ناشی از آن هستند، بر روی چنین پلتفرم‌هایی امکان‌پذیر نیست یا هزینه بسیار زیادی را متحمل می‌شود. یکی از ساده‌ترین راه‌کارهای موجود می‌تواند کاهش حجم مدل‌های شبکه عصبی با تغییر پارامترهای شبکه، مثل کاهش ابعاد ورودی، باشد به طوری که حداقل افت صحت یا افزایش خطا رخ دهد. اما این روش نیازمند تغییرات به ازای شبکه‌ها و مدل‌های مختلف است.

یکی دیگر از روش‌های بسیار پرکاربرد و کم خطا، استفاده از کوانتیزاسیون[۱۲] است. کوانتیزاسیون فرآیندی است که مقادیر پیوسته را به مجموعه محدودی از مقادیر گسسته یا سطوح تقریب می‌دهد یا گرد می‌کند. این فرآیند به طور معمول در پردازش سیگنال دیجیتال

---

[۹] Dataset
[۱۰] projection
[۱۱] Principal component analysis
[۱۲] Quantization

و فشرده‌سازی داده‌ها برای نمایش و ذخیره سازی داده‌ها با کارآیی بیشتر استفاده می‌شود. در مدل‌های شبکه عصبی، عموما پارامترهای وزن مدل را از اعداد ممیز شناور[۱۳] ۳۲ بیت به اعداد ممیز ثابت[۱۴] ۱۶ بیت یا کمتر گرد می‌کنند؛ به عنوان مثال، با تبدیل وزن‌ها به هشت بیت ممیز ثابت، حجم مدل تا چهار برابر کاهش یافته، که مقدار قابل توجهی است [4].

۲- پیش نیازهای انجام تمرین

۱. آشنایی اولیه با پایتان
۲. نرم افزار شبیه سازی زبان توصیف سخت افزار مانند Modelsim

۳- مراحل انجام تمرین

در این تمرین هدف ساخت رمزگذار خودکار با حداقل حجم ممکن و حفظ خطای مدل تا حد امکان، و در نهایت شبیه‌سازی آن با زبان توصیف سخت‌افزار (در اینجا سیستم وریلاگ) است. بنابراین تمرین شامل دو مرحله اصلی است؛ در مرحله اول با شبکه عصبی و آموزش آن و دو نمونه از تکنیک‌های فشرده‌سازی آشنا می‌شوید. در مرحله دوم نیز خروجی مدل نهایی بدست آمده را با زبان توصیف سخت‌افزار سیستم وریلاگ انجام می دهید و شبیه‌سازی خواهید کرد.

۳-۱- مرحله اول:

دو فایل کد پایتان در اختیار شما قرار داده شده است:

۱. فایل hw-mnist.ipynb مربوط به طراحی کدگذار خودکار برای داده‌های MNIST، آموزش و بررسی خروجی آن، کوانتیزاسیون مدل به ۸ بیت ممیز ثابت و در نهایت ذخیره سازی وزن‌های مدل جهت استفاده در زبان توصیف سخت‌افزار.
۲. فایل hw-notmnist.ipynb مربوط به طراحی کدگذار خودکار برای داده‌های NotMNIST مشابه فایل قبل.
۳. لینک دیتاست NotMNIST:
https://drive.google.com/file/d/1OHKWbVBt8lgjmJdCcfZ_JEIpRoxqoaiv/view?usp=sharing

موارد زیر را انجام دهید:

۱) ابتدا فایل اول را اجرا کنید و از متغیر history تعریف شده (که جزئیات آموزش شبکه را نشان میدهد)، نمودار loss را برای داده‌های آموزش و ارزیابی رسم کنید. سپس نمودارها و شکل‌های حاصل در هنگام اجرای کد را گزارش و تحلیل کنید.
۲) تابعی بنویسید که با دریافت اطلاعات یک مدل از ورودی (تعداد نورون‌های هر لایه و تعداد بیت)، حجم نهایی مدل را بر حسب KB، با توجه به ماتریس وزن هر لایه، محاسبه کند، سپس حجم مدل اصلی و مدل کوانتایز شده را گزارش کنید و میزان فشرده‌سازی ایجاد شده را بیان کنید.

نکته: در قسمت آخر فایل کوانتیزاسیون را مشاهده خواهید کرد. در کوانتیزاسیون، بیت علامت جدا از بیت‌های اصلی در نظر گرفته می‌شود؛ به عنوان مثال عدد ۸ بیتی با ۷ بیت اعشار دارای یک بیت اضافه تر به عنوان بیت علامت است، بنابراین ۹ بیت برای کل آن در نظر گرفته می‌شود. به همین جهت است که برای کوانتایز کردن داده‌های ورودی شبکه (x_train و x_test)، ۱۷ بیت (که در واقع ۱۶ بیت نشان دهنده‌ی مقدار عدد و یک بیت مربوط به بیت علامت است) با ۱۴ بیت اعشار در نظر گرفته شده است.

۳) در فایل دوم، کدهای مربوط به خواندن و پیش پردازش تصاویر NotMNIST وجود دارند. مانند MNIST، شبکه‌ی اتوانکدری طراحی کنید و آموزش دهید. تعداد لایه‌های رمزگذار و رمزگشا را حداقل دو و حداکثر چهار در نظر بگیرید.

[۱۳] Floating point
[۱۴] Fixed Point

سپس ۱۵ نمونه خروجی تصاویر از این اتوانکدر را گزارش و تحلیل کنید (به همراه نمودار loss). خطای بدست آمده نباید از ۰/۰۱۱ بیشتر باشد.

۴) یکی از راهکارهای کاهش تعداد پارامترهای شبکه، کاهش اندازه ورودی است؛ زیرا در این صورت می‌توان تعداد نورون‌های کمتری برای لایه‌های میانی در نظر گرفت به طوری که خروجی مطلوب همچنان حاصل شود. سعی کنید با استفاده از تابع داده شده resize_images، اندازه تصاویر را تا حد امکان کاهش دهید و به تبع آن پارامترهای شبکه را تغییر دهید به طوری که تعداد پارامترهای آن حداقل شود. توجه کنید که اختلاف خطای مدل جدید بدست آمده با مدل اولیه، نباید بیش از ۰/۰۰۱ باشد. در نهایت خطا و ۱۵ نمونه خروجی تصاویر بدست آمده از مدل را گزارش کنید.

۵) نمودار هیستوگرام یکی از نمودارهای مفیدی است که می‌تواند بازه اعداد وزن‌ها و تعداد تکرار هریک را برای هر لایه یا ورودی‌ها نشان دهد و به تعیین تعداد بیت برای کوانتایز نمودن مدل کمک کند. نمودار هیستوگرام برای وزن‌های هر لایه را گزارش کنید. به نظر شما بهترین تعداد بیتی که می‌توان برای کوانتایز کردن وزن ها بدون افت دقت در نظر گرفت، به ازای هر لایه چند بیت است؟ برای ورودی ها و خروجی‌ها چطور؟ بر این اساس شبکه را کوانتایز کنید و آموزش دهید و خطای آن و خروجی ۱۵ تصویر را به عنوان نمونه گزارش کنید.

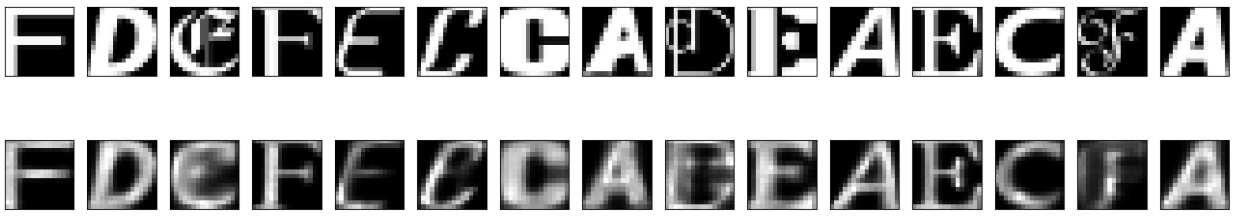
۶) بار دیگر مدل را با شش بیت کوانتایز کنید (به غیر از ورودی و خروجی) و آموزش دهید. سپس خطا و خروجی ۱۵ تصویر نمونه را گزارش کنید.

۷) مرحله ۶ را با چهار بیت نیز تکرار کنید، خطا و خروجی ۱۵ تصویر نمونه را گزارش کرده و با خروجی های موارد ۵ و ۶ مقایسه و تحلیل کنید.

۸) دو نمونه از ورودی‌های کوانتایز شده و همه وزن‌های کوانتایز شده بخش کدگذار از قسمت ۵ را در فایل‌های txt مانند فایل MNIST داده شده ذخیره کنید.

۹) مرحله ۸ را با خروجی مرحله ۷ مقایسه کنید.

در شکل 4 نمونه ای از یک خروجی قابل قبول (فشرده شده) نشان داده شده است.



شکل ۴- نمونه ای از خروجی قابل قبول برای حداقل فشرده سازی

نمره اضافی

از رمزگذار خودکار آموزش داده شده در فایل hw_mnist برای ۱۵ نمونه از تصاویر NotMNIST استفاده کنید و خروجی آن را گزارش کنید. آیا به نظر شما خروجی آن قابل قبول است؟ اگر خیر، آیا می توان بدون نیاز به آموزش مجدد از ابتدا برروی همه دیتاست NotMNIST روشی را پیش گرفت که به نتایج قابل قبولی بتوان دست یافت؟

۲-۳- مرحله دوم:

در این قسمت با نحوه پیاده‌سازی مدل بدست آمده از مرحله قبل آشنا می‌شوید. یک فایل سیستم وریلاگ با اسم node.sv به شما داده شده است. این فایل پیاده‌سازی یک نورون از لایه MLP است که درواقع عملیات MAC را انجام می‌دهد؛ یعنی هریک از ورودی‌های آن نورون را در وزن متناظر خود ضرب می کند و در نهایت با مقدار بایاس جمع می‌کند. برای سادگی، زمان انجام عملیات یک clock

cycle در نظر گرفته شده است. بنابراین فرض بر این است که خروجی هر لایه پس از یک clock cycle به طور کامل آماده خواهد بود. توجه کنید که بیت علامت باید جدا در نظر گرفته شود، به عنوان مثال عدد ۸ بیتی با ۷ بیت اعشاری، درواقع ۹ بیت خواهد بود. همه ماژول‌ها را با سیستم وریلاگ طراحی کنید.

توضیح ورودی‌ها و خروجی‌های ماژول node:

پارامتر LAYERID: شماره لایه‌ای که نورون در آن قرار دارد

پارامتر NODEID: شماره نورون در یک لایه

پارامتر WIDTH: عرض بیت هر یک از ورودی های نورون

پارامتر INPUT_NUM: تعداد ورودی های نورون

پارامتر OUTPUT_WIDTH: عرض بیت خروجی نورون

clk: سیگنال یک بیتی نشان دهنده وضعیت clock

in_ready: ورودی یک بیتی نشان دهنده آماده بودن ورودی ماژول

X: مجموعه‌ی ورودی‌های نورون که در INPUT_NUM * WIDTH بیت قرار گرفته اند

out: خروجی ماژول با عرض بیت OUTPUT_WIDTH بیت

out_ready: خروجی یک بیتی نشان دهنده حاضر بودن خروجی

۱) **ماژول relu:** همانطور که در مدل دیدید، از تابع فعال‌ساز relu برای هر لایه استفاده شده است. این تابع فقط ورودی‌های مثبت را فیلتر می‌کند؛ به طوری که اگر خروجی منفی باشد، صفر بازگردانده شده و در غیر اینصورت همان عدد در خروجی ظاهر می‌شود. ماژولی به اسم ReLu طراحی کنید که در یک clock cycle عملیات مورد نظر را انجام دهد. این ماژول باید شامل ورودی‌ها و خروجی‌های زیر باشد:

clk: سیگنال یک بیتی نشان دهنده وضعیت clock

valid: ورودی یک بیتی نشان‌دهنده معتبر بودن ورودی

x: ورودی اصلی به ماژول

out: خروجی ماژول

out_ready: خروجی یک بیتی نشان دهنده حاضر بودن خروجی ماژول

۲) **ماژول clip:** این ماژول ورودی w بیتی را دریافت می کند و خروجی out_w بیتی را برمی‌گرداند؛ بدین صورت که ورودی را با بیشترین و کمترین عدد در out_w مقایسه می کند، اگر بیشتر از این بازه باشد، بیشترین عدد آن، اگر کمتر باشد، کمترین عدد آن و در غیر اینصورت خود ورودی را برمیگرداند.

راهنمایی: در صورتی که ورودی در بین بازه مورد نظر با out_w بیت باشد، باید خروجی به صورت زیر انتخاب شود (عدد ۱۷ بیتی زیر با ۱۴ بیت اعشار را در نظر بگیرید):

00110110001011001

Integer part: **0**01

Fraction part: 10110001011001

در صورتی که خروجی مورد نظر ۹ بیت با ۷ بیت اعشار باشد، از بخش صحیح بیت اول از راست به همراه بیت علامت آن را جدا می کنیم و برای بخش صحیح خروجی قرار می‌دهیم. برای بخش اعشاری نیز از سمت چپ به تعداد مورد نیاز یعنی ۷ بیت را جدا می کنیم و به عنوان بخش اعشاری خروجی قرار میدهیم:

01 10110001

این ماژول باید شامل ورودی‌ها و خروجی‌های زیر باشد:

clk: یک ورودی یک بیتی clock

valid: ورودی یک بیتی نشان‌دهنده معتبر بودن ورودی

x: ورودی اصلی به ماژول

out: خروجی ماژول

out_ready: خروجی یک بیتی نشان دهنده حاضر بودن خروجی ماژول

۳) **ماژول layer**: در این ماژول به تعداد لازم از نورون، ReLu و clip نمونه‌سازی می‌شود. این کار را با استفاده از generate در verilog و یک حلقه for انجام دهید. این ماژول باید شامل ورودی‌ها و خروجی‌های زیر باشد:

clk: یک ورودی یک بیتی clock

valid: ورودی یک بیتی نشان‌دهنده معتبر بودن ورودی

x: ورودی اصلی به ماژول (دقیقاً مشابه ورودی به هر نورون)

out: خروجی ماژول که یک آرایه از خروجی‌ها به ازای هر نورون است.

out_ready: خروجی یک بیتی نشان دهنده حاضر بودن خروجی ماژول

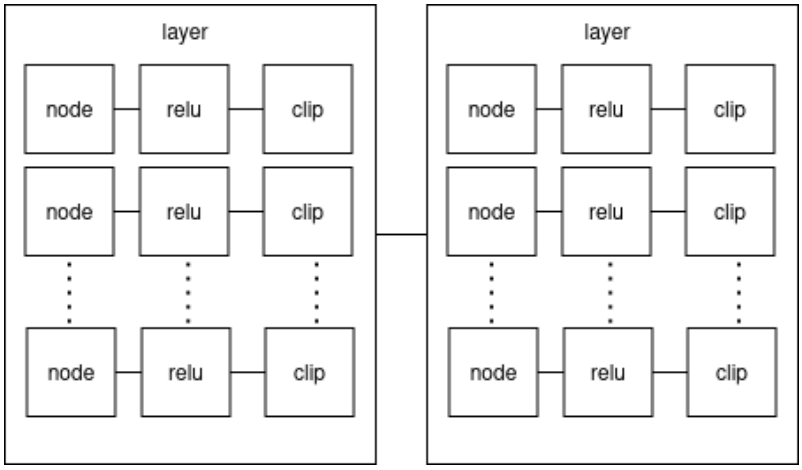
۴) در نهایت در یک فایل encoder.sv، به تعداد لایه‌های کدگذار نهایی که وزن‌های آن را ذخیره کردید، از ماژول layer نمونه‌سازی کنید و با متصل کردن ورودی و خروجی‌های آن‌ها به یکدیگر به شکل مناسب، خروجی ها را تولید کنید.

۵) یک فایل به نام testbench.sv ایجاد کنید و در آن از ماژول encoder یک نمونه بسازید. در این ماژول باید در بخش initial فایل ورودی ذخیره شده از مرحله قبل(کد پایتان) را خوانده و به عنوان ورودی لایه اول در رجیستر مناسب ذخیره کنید.

۶) نتایج شبیه‌سازی را گزارش کنید و توضیح دهید.

۷) خروجی‌های تولید شده را در یک فایل out.txt به صورت دسیمال ذخیره کنید. سپس آن‌ها را در یک کد پایتان از فایل خوانده و به عنوان ورودی به کدگشای طراحی شده وارد نموده و خروجی تصویر آن را با خروجی بدست آمده از تصویر مرحله اول مقایسه کنید.

شکل ۵ شمایی از معماری که باید پیاده‌سازی شود را نشان می‌دهد.



شکل ۵- شمای کلی معماری مورد نیاز

نمره اضافی

برای بخش رمزگشای بدست آمده از مرحله اول(کد پایتان) نیز پیاده‌سازی آن را انجام دهید و نام ماژول آن را decoder بگذارید. این دو ماژول را به هم متصل کنید و نام آن را denoiser بگذارید. سپس در یک فایل testbench، از آن روی دو نمونه از ورودی‌ها، خروجی شبیه‌سازی و فایل خروجی را بدست آورید و با خروجی‌های بدست آمده از مراحل اول مقایسه کنید.

لازم است موارد زیر جهت تحویل تمرین و ارائه‌ی گزارش رعایت شوند:

- گزارش خود را در بخش‌های مجزا شامل چکیده، نحوه‌ی انجام کار، نتایج به دست آمده، تحلیل نتایج، نتیجه‌گیری و ضمائم بیاورید. فایل گزارش باید بر اساس فرمت قرار داده شده در سایت درس باشد.
- در پیاده‌سازی های سخت‌افزاری، لازم است همه ماژول‌ها را تا حد امکان پارامتری بنویسید (به عنوان مثال تعداد نورون‌های ورودی لایه، عرض بیت و ... پارامتر باشند).
- درصورت استفاده از تکنیک‌های اضافه برای فشرده‌سازی، در گزارش توضیح دهید.
- فایل گزارش به صورت doc باشد. کد خود را نیز آپلود کنید.
- تمرین را با فرمت YourName_StudentNo_EAI1.rar آپلود کنید.
- گروه ها حتما دو نفره باشند.
- بارگذاری فایل‌های گزارش توسط یکی از اعضای گروه کافی است.
- نمره از 100 محاسبه می شود و به ازای هر روز تاخیر در اپلود تمرین، به اندازه 2^x (x تعداد روز تاخیر) از نمره شما کسر می‌شود.
- در صورت مشاهده تشابه زیاد در کدها و گزارش، نمره 100- برای هر دو گروه اعمال خواهد شد.
- تمرین تحویل حضوری دارد که زمان آن بعدا اعلام خواهد شد.

۴- مراجع

[1] Available online at: http://yann.lecun.com/exdb/mnist/
[2] Available online at: https://www.kaggle.com/datasets/jwjohnson314/notmnist
[3] https://blog.keras.io/building-autoencoders-in-keras.html
[4] Krishnamoorthi, R., 2018. Quantizing deep convolutional networks for efficient inference: A whitepaper. arXiv preprint arXiv:1806.08342.