VISION

بسم تعالی

مبانی علوم اعصاب

پروژه بررسی و مدلسازی سیستم بینایی به وسیله تسک رفتاری و مدل HMAX

امیرحسین زاهدی ۹۹۱۰۱۷۰۵

پاییز ۱۴۰۲

فهرست

چکیده

برای بررسی ساختار Feed forwarding در مسیر بینایی انسان یک مدل محاسباتی ارائه شده است (HMAX) که بر پایه محاسبات سلسله مراتبی عمل می کند. این مدل برای شبیه‌سازی مسیر پردازش بصری ونترال در قشر پستانداران طراحی شده است. مدل مطرح شده یک فرآیند یادگیری دو مرحله‌ای را شامل می‌شود، از جمله یک مرحله یادگیری بدون نظارت آهسته شبیه به رشد و پس از آن آموزش ویژه کار. این مدل با الهام از داده‌های زیست‌شناسی عصبی، با آناتومی و فیزیولوژی قشر بینایی، با در نظر گرفتن مسیرهای بای‌پس و ترکیب مقادیر پارامترهای کمی، همسو می‌شود.

این مدل عملکرد چشمگیری را در تشخیص اشیاء در صحنه های طبیعی، نشان دادن استحکام نسبت به تغییرات پارامترها و شکل دقیق عملیات اساسی نشان می دهد. از نزدیک مطالعات فیزیولوژیکی را تحت محرک های فیلتر شده تقلید می کند، که نقش مهمی را برای اجزای پیشروی پایین به بالا (Feed forward) در پاسخ های سلول های قشر مغز را نشان می دهد. این مطالعه قابلیت‌های مدل را در یک کار دسته‌بندی فوق‌العاده سریع، به‌ویژه یک کار تشخیص حیوان در مقابل غیرحیوان، که در آن مدل و ناظران انسانی عملکرد مشابهی از خود نشان می‌دهند، ارزیابی می‌کند. همبستگی بالای مدل با ناظران انسانی در دسته‌های مختلف تصویر و توانایی آن در پیش‌بینی رفتار انسان‌مانند در موقعیت‌هایی مانند چرخش تصویر، اثربخشی آن را برجسته می‌کند. با توجه به مطالعات قبلی نتایج انسان و مدل تقریبا یکسان هستند اما در مطالعه انجام شده در این بررسی، انسان از مدل عملکرد بهتری از خود نشان داده است.

با این حال، این مطالعه محدودیت‌های یک معماری کاملاً Feed forward را تأیید می‌کند، به‌ویژه در مدیریت تصاویر به هم ریخته. الحاقات اولیه به مدل اهمیت پیش بینی های feed back را در بهبود عملکرد در شرایط به هم ریخته نشان می دهد. علی‌رغم این محدودیت‌ها، یافته‌ها پشتیبانی علوم اعصاب محاسباتی را برای وجود یک مرحله یادگیری مستقل، بدون نظارت و رشد مانند در مسیر ونترال فراهم می‌کنند. این مدل، مطابق با فیزیولوژی و آناتومی شناخته شده، بینش‌های ارزشمندی را در مورد مراحل اولیه پردازش بصری ارائه می‌دهد و زمینه را برای درک وظایف بصری پیچیده‌تر با واسطه‌گری برآمدگی‌های مکرر در قشر بینایی فراهم می‌کند.

مقدمه

در مطالعه ای که در کلاس درس مبانی علوم اعصاب مورد بررسی قرار گرفت، محققان استفاده از روش Feed forward را برای وظایف طبقه بندی سریع تصاویر در مغز مورد بررسی قرار دادند. آنها دریافتند که با توجه به تاخیر هایی که در مسیر پردازش بینایی می تواند رخ بدهد، توانایی تشخیص عکس ها سریع تر از تاخیر های مسیر، ناشی از شیوه Feed forward مغز در تشخیص اولیه تصاویر است. همچنین آن ها به ساختار سلسله مراتبی بینایی نیز پی بردند به صورتی که هر چه به مراحل بالاتر می رویم، ناحیه مورد بررسی توسط نورون ها بزرگتر می شود و پترن های پیچیده تری تحلیل می شود.

آن ها قصد داشتند تا با یک پیاده سازی خاص از این معماری قادر باشند سطح و الگوی عملکرد به دست آمده توسط انسان را بر روی تصویر یک حیوان فیلتر شده را در مقابل تصاویر غیر حیوان فیلتر شده پیش بینی کنند. برای طبقه بندی حیوانات محققان از یک رویکرد نورومورفیک برای بینایی کامپیوتری استفاده کردند که شامل درک مکانیسم های مغزی زیربنایی تشخیص اشیا و صحنه های بصری پیچیده است.

به طور کلی، این مطالعه نشان می‌دهد که یک معماری Feed forward می‌تواند ابزار مؤثری برای دسته‌بندی سریع باشد. معماری مورد استفاده در این مطالعه قادر به دستیابی به سطحی از عملکرد قابل مقایسه با ناظران انسانی بود و می‌توانست سطح و الگوی عملکرد به دست آمده توسط انسان را در تصویر حیوان فیلتر شده را پیش‌بینی کند. . این نوع معماری کاربردهای بالقوه ای در زمینه های مختلف از جمله تشخیص تصویر، تشخیص گفتار و پردازش زبان طبیعی دارد.

در این پروژه قصد داریم که سیستم تشخیص و طبقه بندی سریع تصاویر توسط مغز را که گمان می شود بر پایه روش Feed forward عمل می کند را به وسیله یک تسک رفتاری مورد بررسی قرار دهیم. پس از آن مدلسازی انجام شده بر پایه بینایی انسان را که مدل HMAX نام دارد را مورد بررسی قرار می دهیم و به لحاظ عملکردی با نتایج بدست آمده از تسک رفتاری مقایسه می کنیم. در طراحی تسک از تولباکس psychtoolbox متلب استفاده شده است.

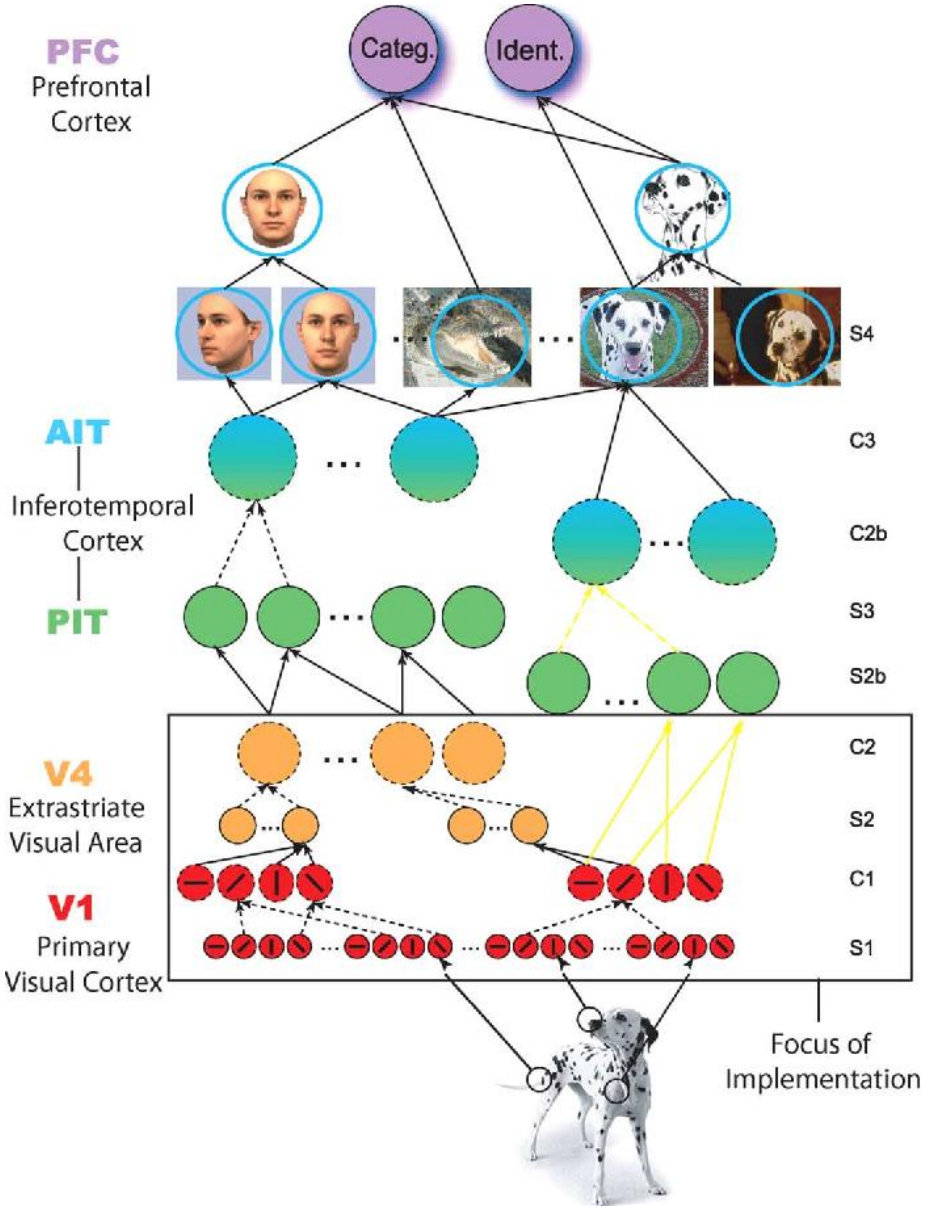
مدل HMAX از تلاش‌ها برای پر کردن شکاف بین علوم اعصاب محاسباتی و بینایی رایانه با ارائه یک چارچوب بیولوژیکی قابل قبول برای تشخیص بصری ناشی می‌شود. HMAX با الهام از سازماندهی سلسله مراتبی نواحی قشر بینایی، جنبه های کلیدی مسیر ونترال را که مسئول تشخیص اشیا در پستانداران است، ثبت می کند. این مدل سلول های ساده و پیچیده، ادغام فضایی و جهت گیری، و سلسله مراتبی از مراحل پردازش را در بر می گیرد که منعکس کننده روش پردازش اطلاعات در سیستم بصری است.

در حوزه روان‌شناسی، محققان از مدل HMAX برای بررسی شباهت‌ها و تفاوت‌های پردازش بصری بین انسان و حیوانات غیرانسان استفاده کرده‌اند. سایکوفیزیک شامل مطالعه رابطه بین محرک‌های فیزیکی و پاسخ‌های روان‌شناختی است که آنها برمی‌انگیزند، و مدل HMAX در روشن کردن مکانیسم‌های عصبی نهفته در ادراک بصری ارزشمند است.

این گزارش به بررسی کاربرد مدل HMAX در مطالعات سایکوفیزیکی با مقایسه وظایف تشخیص بصری حیوانات و غیرحیوانی می‌پردازد. با بررسی نحوه عملکرد این مدل در گونه‌های مختلف، هدف محققان دستیابی به بینش‌هایی در مورد جهانی بودن یا خاص بودن مکانیسم‌های پردازش بصری است. علاوه بر این، ادغام آزمایش‌های روان‌فیزیکی با مدل

HMAX چارچوبی برای درک اساس عصبی تشخیص شی در سیستم‌های بیولوژیکی و مصنوعی فراهم می‌کند. این تحقیق به درک گسترده‌تر ما از شناخت بصری کمک می‌کند و ممکن است پیامدهایی برای توسعه سیستم‌های بینایی رایانه‌ای با الهام از کارایی پردازش بصری بیولوژیکی داشته باشد.

مدل محاسباتی معرفی شده یک تصویر با ارزش خاکستری را از طریق سلسله مراتبی از واحدهای ساده (S) و پیچیده (C) پردازش می کند. واحدهای S۱ به میله‌ها و لبه‌های جهت‌دار پاسخ می‌دهند، در حالی که واحدهای C۱، مشابه سلول‌های مخطط پیچیده، خروجی‌های واحدهای S۱ را برای دستیابی به عدم حساسیت نسبت به مکان و مقیاس جمع‌آوری می‌کنند. این مدل این سلسله مراتب را به مناطق خارج از مرز گسترش می دهد و به یک مبادله بین انتخاب پذیری و تغییر ناپذیری دست می یابد. در هر مرحله، واحدهای ساده با ویژگی‌های پیچیدگی فزاینده تنظیم می‌شوند، در حالی که واحدهای پیچیده، آوران‌ها را با گزینش‌های متفاوت ادغام می‌کنند و پیچیدگی نمایش را افزایش می‌دهند. هدف این مدل محاسبه کمی داده های تشریحی و فیزیولوژیکی در قشر بینایی است.



روش آزمایش: تسک رفتاری

تسک طراحی شده برای بررسی الگوی طبقه بندی سریع توسط مغز، تشخیص تصویر حیوان از تصویر غیر حیوان است که در محیط متلب و با استفاده از تولباکس psych پیاده سازی شده است.

در این تسک در هر مرحله تصویری با مدت زمان بسیار کوتاه (۲۰ میلی ثانیه) به فرد مورد آزمایش نمایش داده می شود و او باید تصمیم بگیرد که تصویر مشاهده شده حیوان یا غیر حیوان بوده است.

در این آزمایش از ۶۰۰ عکس حیوان در ۴ دسته سر، فاصله نزدیک از حیوان، فاصله متوسط از حیوان و فاصله دور از حیوان و همچنین از ۶۰۰ عکس از هر وسیله غیر حیوان در همین ۴ دسته که در دیتا ست مقاله بیان شده موجود بودند استفاده شده است.

افراد مورد آزمایش یک مرحله ترینینگ و دو مرحله تست را پشت سر می گذارند. در هر مرحله از ۴۰۰ عکس استفاده شده است که ۲۰۰ تای آن ها حیوان و ۲۰۰ تا غیر حیوان هستند. در ۲۰۰ عکس حیوان نیز هر دسته نوع عکس، حاوی ۵۰ تصویر است.

پس از مقدماتی که درباره تسک طرحی شده مطرح شد، مراحل پیاده سازی تسک رفتاری را بیان می کنیم.

تابع تسک طراحی شده در فایل AnimalOrnot.m موجود است. این تابع سه ورودی دارد که در ورودی اول نام فرد، در ورودی دوم شماره ترایل و در نهایت تعداد عکس مورد آزمایش در آن ترایال به عنوان ورودی داده می شود:

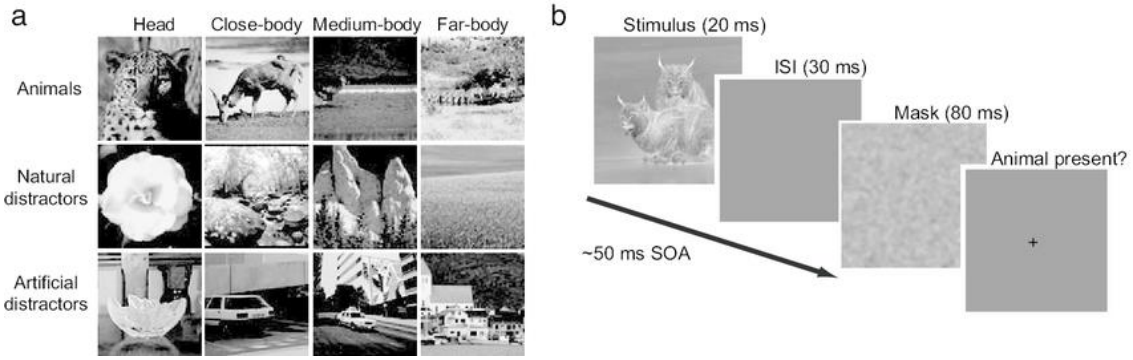
```
function AnimalOrnot(subject_name,n,image_counts)
```

در ابتدا تصاویر را به صورت دستی در ۸ دسته دسته بندی می کنیم که ۴ تای آن ها حیوان و ۴ تا غیر حیوان هستند. سپس این ۸ دسته را در ۸ سلول تعریف شده در متلب لود ایمپورت می کنیم. برای مثال قطعه کد بارگزاری عکس های مربوط به دسته (حیوان،سر) به صورت زیر است.

```
%%%%%%%%%%%%%%%%%%%% Data reading %%%%%%%%%%%%%%%%%%%%
    img_dir = 'H_a\';
    file_list = dir([img_dir '*.jpg']);
    img_H_a = cell(150, 1);
    for i = 1:length(file_list)
        filename = [img_dir file_list(i).name];
        img_H_a{i} = imread(filename);
    end
```

پس از آنکه بارگزاری تصاویر انجام شد، سه ترایال می سازیم که هر کدام شامل ۴۰۰ عکس است. برای مثال ساخت ترایال اول که مربوط به ترینینگ است مشاهده می شود: (از هر دسته ۵۰ عکس در ترایال قرار داده می شود.)

```
%%%%%%%%%%%%%%%%%%% Trial making %%%%%%%%%%%%%%%%%%%
trial1 = cell(400,1);
trial1(1:50) = img_H_a(1:50);
trial1(51:100) = img_H_d(1:50);
trial1(101:150) = img_C_a(1:50);
trial1(151:200) = img_C_d(1:50);
trial1(201:250) = img_M_a(1:50);
trial1(251:300) = img_M_d(1:50);
trial1(301:350) = img_F_a(1:50);
trial1(351:400) = img_F_d(1:50);
```



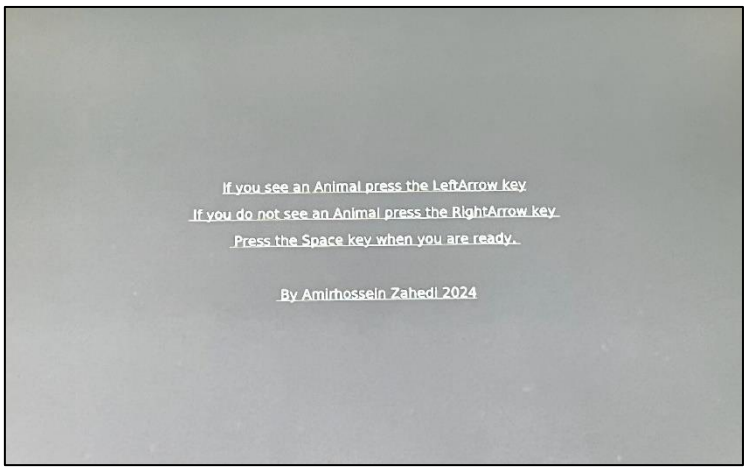
حال با توجه به تعداد عکس داده شده و شماره ترایال توسط کاربر، ترتیب رندوم عکس ها در ترایال مورد نظر تولید می شود تا با استفاده از این ترتیب رندوم، عکس ها برای سابجکت پخش شوند.

```
%%%%%%%%%%%%%%%%%%% Trial defining %%%%%%%%%%%%%%%%%%%
    trial_number = image_counts;
    random_im = randperm(400, 400);
    switch n
        case 1
            trial = trial1;
        case 2
            trial = trial2;
        case 3
            trial = trial3;
        otherwise
            trial = trial1;
    end
```

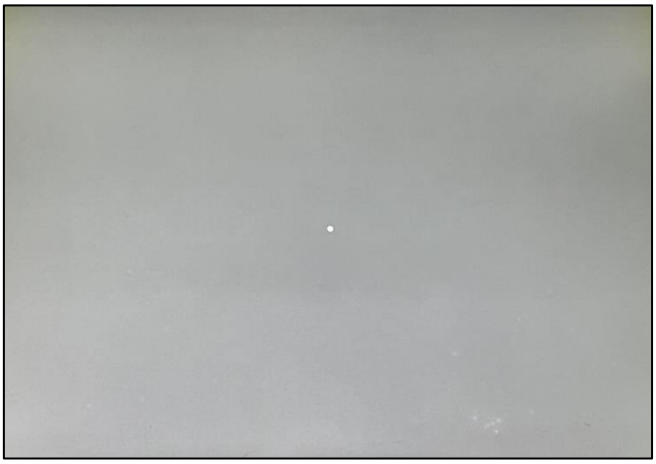
پس از مشخص شدن مطالب مربوط به دیتاست، به سراغ مراحل پیاده سازی تسک به صورت تصویری می رویم. پارادایم آزمایش به صورت زیر است.

در ابتدای تسک توضیحات اولیه به صورت تصویری ارائه می شود. سپس با فشردن اسپیس، نقطه ای سفید بر روی مرکز تصویر نمایش داده می شود و سپس بعد از دیلی کوتاه، تصویر محرک به مدت ۲۰ میلی ثانیه (۱۸ میلی ثانیه طب محاسبه) پخش می شود.

```
%%%%%%%%%%%%%%%%%%%% First scene %%%%%%%%%%%%%%%%%%%%
message = sprintf('If you see an Animal press the LeftArrow key\n\n If you do not
see an Animal press the RightArrow key \n\n Press the Space key when you are
ready. \n\n\n\n By Amirhossein Zahedi 2024');
DrawFormattedText(window, message, 'center','center', [256,256,256]);
HideCursor;
Screen('Flip', window);
```




```
%%%%%%%%%%%%%%%%%%%% Center scene %%%%%%%%%%%%%%%%%%%%
t_center1 = GetSecs;
Screen('DrawDots', window, [xCenter, yCenter], 20, [1 1 1], [], 2);
Screen('Flip', window);
Screen('Flip', window, t_center1 + (30) * ifi);
```



```
%%%%%%%%%%%%%%%%%%%% Delay for image showing %%%%%%%%%%%%%%%%%%%%
delay = randi([10,20]);
t_delay = GetSecs;
Screen('Flip', window);
Screen('Flip', window, t_delay + (delay) * ifi);
```

قبل از پخش تصویر ابتدا تصویر مورد نظر را سیاه سفید می کنیم سپس کتگوری عکس و همچنین حیوان بودن یا نبودنش مشخص می کنیم تا طبق پاسخ ثبت شده بتوان اطلاعات را ثبت کرد.

```
%%%%%%%%%%%%%%%%%%%% Image defining & setting %%%%%%%%%%%%%%%%%%%%
image = rgb2gray(trial{random_im(i)});
imageTexture = Screen('MakeTexture', window, image);
if (random_im(i)<=50 && random_im(i)>=1)
            cat = 1;
            animal = 1;
elseif (random_im(i)<=100 && random_im(i)>=51)
            cat = 1;
            animal = -1;
elseif (random_im(i)<=150 && random_im(i)>=101)
            cat = 2;
            animal = 1;
elseif (random_im(i)<=200 && random_im(i)>=151)
            cat = 2;
            animal = -1;
elseif (random_im(i)<=250 && random_im(i)>=201)
            cat = 3;
            animal = 1;
elseif (random_im(i)<=300 && random_im(i)>=251)
            cat = 3;
            animal = -1;
elseif (random_im(i)<=350 && random_im(i)>=301)
            cat = 4;
            animal = 1;
elseif (random_im(i)<=400 && random_im(i)>=351)
            cat = 4;
            animal = -1;
end
```

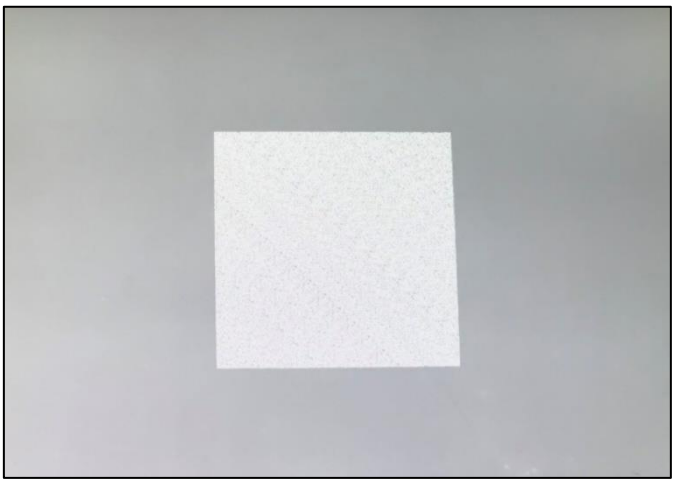
تصویر را به همراه تاخیر بین فیلتر و خود تصویر پخش می کنیم. این مدت زمان میانی که به آن ISI گفته می شود در حدود ۵۰ میلی ثانیه به طول می انجامد.

```
%%%%%%%%%%%%%%%%%%%% Image showing %%%%%%%%%%%%%%%%%%%%
t_image = GetSecs;
Screen('DrawTexture', window, imageTexture, [], [xCenter-400 yCenter-400
xCenter+400 yCenter+400], 0);
Screen('Flip', window);
Screen('Flip', window, t_image + (waitframes_image) * ifi);
```

```
%%%%%%%%%%%%%%%%%%% ISI %%%%%%%%%%%%%%%%%%%
t_ISI = GetSecs;
Screen('Flip', window);
Screen('Flip', window, t_ISI + (waitframes_ISI)* ifi);
```

برای اینکه جلوی مسیر های فیدبکی سیستم بینایی مغز را بگیریم نیاز است که در فاصله زمانی کوتاهی پس از اعمال محرک، محرک نویزی اعمال کنیم تا مسیر های بینایی را پر کند و اجازه ندهد که مغز تحلیل هایی از بالا به پایین انجام دهد. البته این تحریک نباید خیلی زود انجام شود تا تصویر اصلی را نیز با آسیب همراه کند. هدف از این بخش آن است که بتوانیم صرفا مسیر های Feed forward را در کورتکس بینایی به جهت طبقه بندی عکس ها مورد استفاده و بررسی قرار دهیم. نویز را به صورت زیر تولید کرده و سپس نمایش می دهیم.

```
%%%%%%%%%%%%%%%%%%% Noise filter %%%%%%%%%%%%%%%%%%%
noiseid = CreateProceduralNoise(window, 800, 800, 'Perlin', [0.7 0.7 0.7 0.5]);

%%%%%%%%%%%%%%%%%%% Noise showing %%%%%%%%%%%%%%%%%%%
t_filter = GetSecs;
Screen('DrawTexture', window, noiseid, [], [], [], [], [], [], [], [], [1, 0, 0,
0]);
Screen('Flip', window);
Screen('Flip', window, t_filter + (waitframes_filter)*ifi);
```

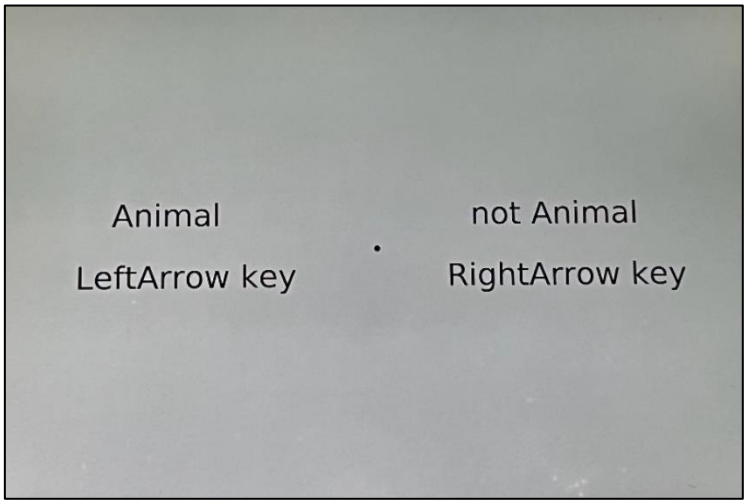


لازم به ذکر است که در ابتدای هر مرحله، زمان شروع را می سنجیم تا با توجه به آن مدت زمان اعمال آن مرحله را بدست آوریم. برای مثال نیاز است که فیلتر نویزی به مدت ۸۰ میلی ثانیه پخش شود و لازم است که زمان پخش فیلتر را بدانیم تا با دلتای ۸۰ میلی ثانیه، زمان پایان را نیز بدست آوریم. به صورت کلی مدت زمان سه مرحله اصلی در این بخش بدست آمده است:

```
%%%%%%%%%%%%%%%%%%% Epoch durations %%%%%%%%%%%%%%%%%%%
showImage_duration = 0.02;
waitframes_image = round(showImage_duration / ifi);
showImage_ISI = 0.03;
waitframes_ISI = round(showImage_ISI / ifi);
showImage_filter = 0.08;
waitframes_filter = round(showImage_filter / ifi);
```

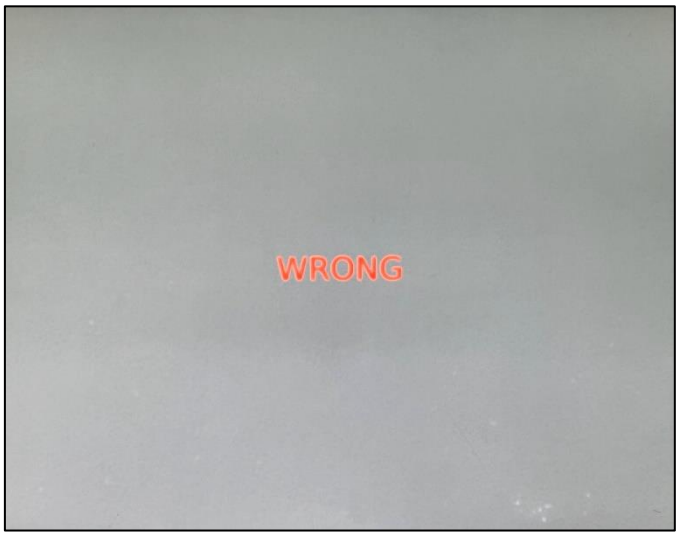
حال که محرک و ISI و نویز را نشان دادیم، نیازمند پاسخ گیری از سمت فرد مورد آزمایش هستیم. در اینجا نقطه ای سیاه بر مرکز تصویر نمایش داده می شود، همچنین در هر سمت نیز پاسخ مورد نظر به همراه کلید متناظرش نوشته شده است. در این بخش فرد باید کلید جهت راست را برای غیر حیوان و کلید جهت چپ را برای حیوان بفشرد. پاسخ او به همرا مدت زمانی که طول کشیده تا پاسخش را ثبت کند، به عنوان مدت زمان واکنش ثبت می شود.

```
%%%%%%%%%%%%%%%%%%%% Response scene %%%%%%%%%%%%%%%%%%%%
Screen('TextSize', window, 100);
Screen('TextStyle', window, 1);
message = sprintf('Animal                          not Animal \n\n LeftArrow key
RightArrow key');
DrawFormattedText(window, message, 'center','center', [0,0,0]);
Screen('DrawDots', window, [xCenter, yCenter], 20, [0 0 0], [], 2);
HideCursor;
Screen('Flip', window);
t_response_start = GetSecs;

while ~(key(1) == kL || key(1) == kR)
            key(1) = nh_key_resp(-1);
end
if (key(1) == kL)
            output(i,3) = 1;
else
            output(i,3) = -1;
end


%%%%%%%%%%%%%%%%%%%% Response time %%%%%%%%%%%%%%%%%%%%
t_response_end = GetSecs;
response_time = t_response_end - t_response_start;
output(i,4) = response_time;
```



اگر ترایال اول که مربوط به ترینینگ است انجام شود، پس از هر ریسپانس، درست یا غلط بودن گزینه انتخاب شده به فرد نمایش داده می شود تا درست یا غلط بودن انتخاب خود را به جهت آموزش بفهمد. البته این قابلیت برای ترایال های ۲ و ۳ که برای تست هستند وجود ندارد.

```
%%%%%%%%%%%%%%%%%%%% Training scene (Trial 1) %%%%%%%%%%%%%%%%%%%%
if n == 1
       if output(i,3) == animal
              message = sprintf('CORRECT');
              DrawFormattedText(window, message, 'center','center', [0,256,0]);
              HideCursor;
       else
              message = sprintf('WRONG');
              DrawFormattedText(window, message, 'center','center', [256,0,0]);
              HideCursor;
       end
              Screen('Flip', window);
              Screen('Flip', window, t_response_end + (30)* ifi);
       end
```



این آزمایش به تعداد تصاویری که به عنوان ورودی می دهیم (معمولا ۴۰۰) انجام می شود و اطلاعات لازم مربوط به کتگوری هر عکس به همراه حیوان بودن یا نبودن، پاسخ فرد و مدت زمان پاسخ در خروجی Output ذخیره می شود. خروجی به دست آمده توسط تابع measurements.m پردازش می شود. در این تابع میزان دقت فرد به صورت کلی و در هر ۴ کتگوری و همچنین مدت زمان واکنش نیز به صورت کلی و در هر ۴ کتگوری محاسبه و به عنوان خروجی داده می شود.

```
%%%%%%%%%%%%%%%%%%%% Accuracy & RT of full trial %%%%%%%%%%%%%%%%%%%%
    rt_all = 0;
    t = 0;
    for i = 1:image_counts
        if output(i,2) == output(i,3)
            t = t+1;
        end
        rt_all = rt_all + output(i,4);
    end
    accuracy_all = (t/image_counts);
    rt_all = (rt_all/image_counts);
```

```
%%%%%%%%%%%%%%%%%%%% Accuracy & RT of each categury %%%%%%%%%%%%%%%%%%%%
accuracy = zeros(4,1);
t = zeros(4,1);
counts = zeros(4,1);
rts = zeros(4,1);
for i = 1:length(output(:,1))
    if output(i,1) ~= 0
        if output(i,2) == output(i,3)
            t(output(i,1)) = t(output(i,1)) + 1;
        end
        rts(output(i,1)) = rts(output(i,1)) + output(i,4);
        counts(output(i,1)) = counts(output(i,1)) + 1;
    end
end
for i = 1:4
    accuracy(i) = t(i)/(counts(i));
    rts(i) = rts(i)/(counts(i));
end

%%%%%%%%%%%%%%%%%%%% Displays %%%%%%%%%%%%%%%%%%%%
disp("Accuracy of all = " + accuracy_all*100);
disp("RT of all = " + rt_all);
disp("Accuracy of Head = " + accuracy(1)*100);
disp("RT of Head = " + rts(1));
disp("Accuracy of Close Body = " + accuracy(2)*100);
disp("RT of Close Body = " + rts(2));
disp("Accuracy of Medium Body = " + accuracy(3)*100);
disp("RT of Medium Body = " + rts(3));
disp("Accuracy of Far Body = " + accuracy(4)*100);
disp("RT of Far Body = " + rts(4));
```

از تابع user.m نیز برای استفاده کاربری استفاده می شود که در آن مشخصات ورودی توابع داده می شوند.

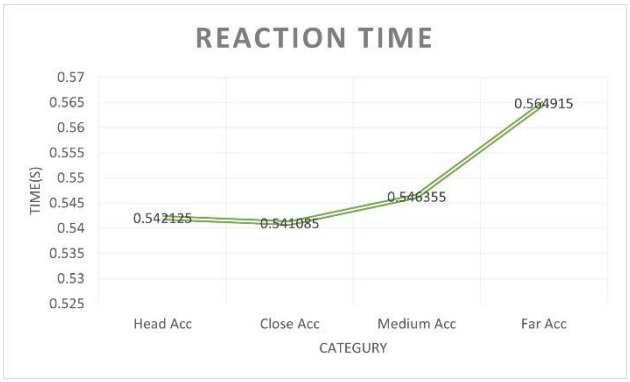
هر سه سابجکت هر کدام یک ترایال ۴۰۰ تصویری را به عنوان تمرین پاسخ می دهند. سپس هر کدام دو ترایال ۴۰۰ تصویری را به عنوان تست پاسخ می دهند. در نهایت پاسخ های ثبت شده آن ها مورد بررسی قرار می گیرد و از دیتای دقت و مدت زمان واکنش بدست آمده از آزمایشات، در بدست آمده نتایج استفاده می گردد.

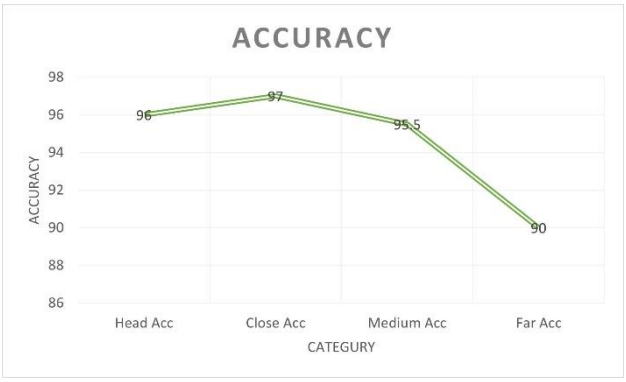
نتایج: تسک رفتاری

سه سابجکت تسک را انجام می دهند که هر کدام ۱ ترایال ۴۰۰ تایی ترینینگ همراه با درستی یا غلطی هر پاسخ می دهد و دو ترایال اصلی تست هر کدام با ۴۰۰ تصویر را آزمون می دهند. از نتایج تست ها میزان دقت و زمان واکنش به صورت کلی و به ازای هر کتگوری سر، نزدیک، متوسط و دور بدست می آید.

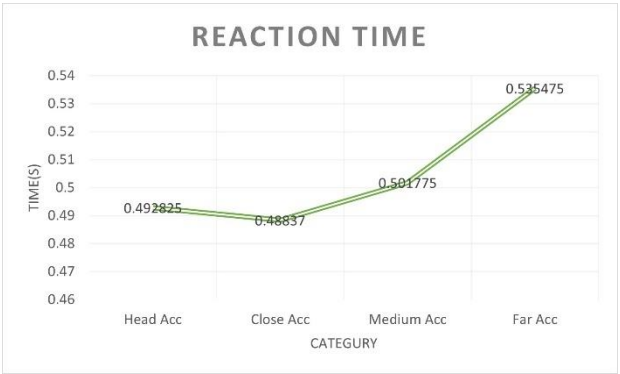
ابتدا نتایج را به تفکیک هر سابجکت نمایش می دهیم:

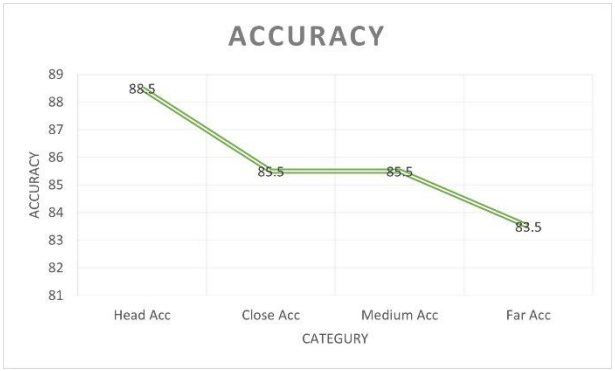
سابجکت اول – امیرحسین:



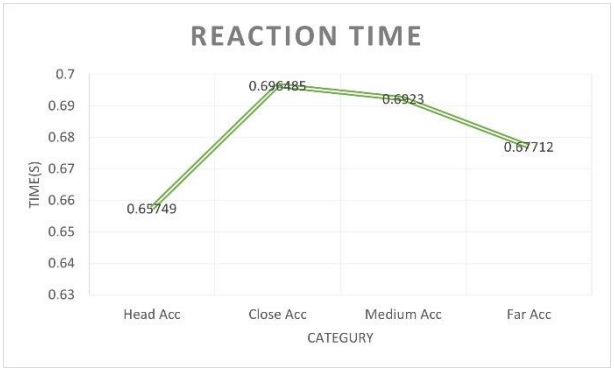


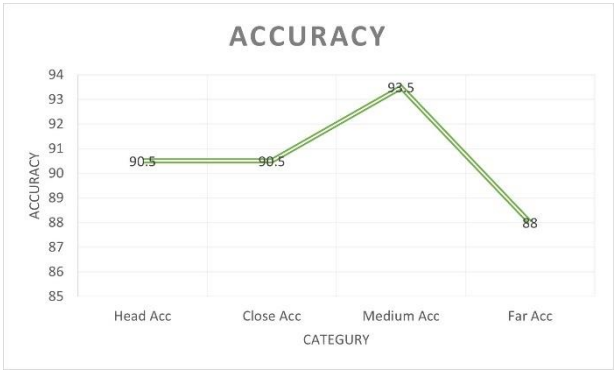
سابجکت دوم – امیرارسلان:





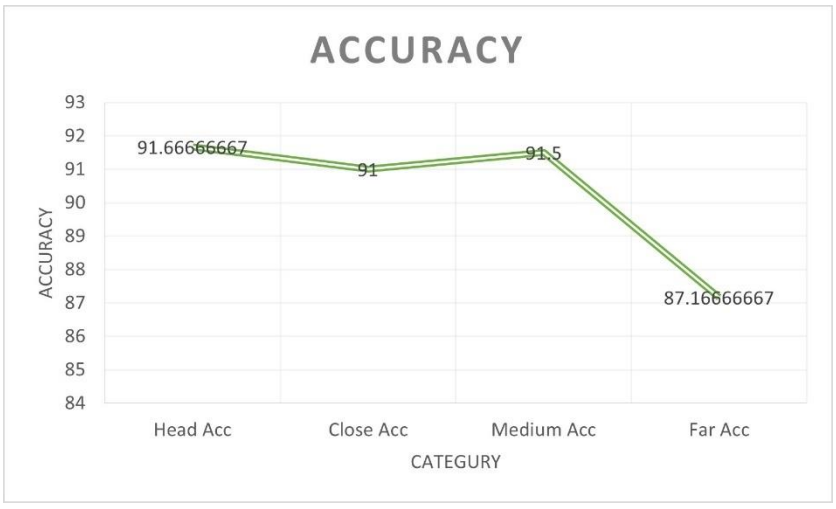
سابجکت سوم – مریم:



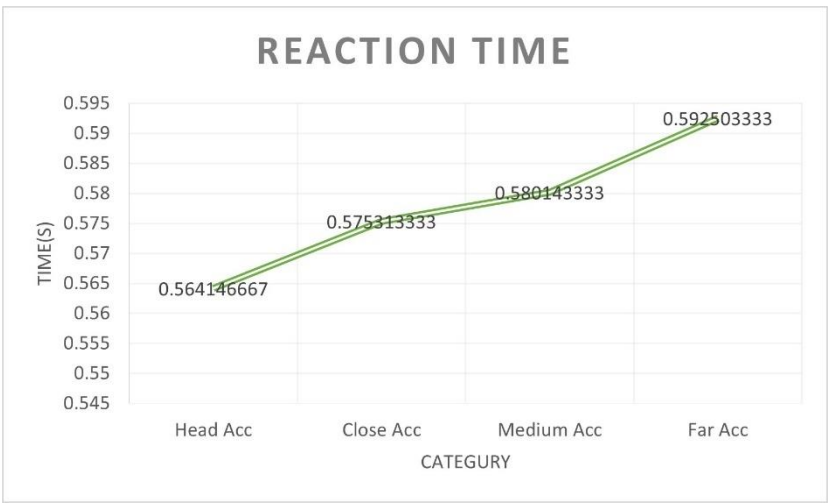


نتایج میانگین و نهایی:

دقت عمل:



زمان واکنش:



همانطور که طبق نتایج دیده می شود، دقت کلی سه سابجکت در تمام ترایال ها به طور میانگین ۹۰.۳۳ درصد است و به صورت میانگین زمان واکنش ۰.۵۷۸ ثانیه است.

اگر بخواهیم هر کتگوری را به صورت جدا گانه ببینیم گویا هر چه تصویر حیوان دور تر می شود، زمان واکنش افزایش می یابد و این افزایش تقریبا خطی است. درباره میزان دقت نیز اگر نگاه کنیم در سه کتگوری اول، دقت ها تقریبا یکسان است اما وقتی تصویر حیوان یا غیر حیوان در دسته دور قرار می گیرد، دقت کاهش پیدا می کند و این کاهش در حدود ۴ درصد است.

نتایج بالا نشان دهنده این است که بدست آوردن اطلاعات و دسته بندی تصاویر حیوانات توسط مغز با استفاده از مسیر های Feed forward به میزان فاصله یا به عبارتی حجمی که در تصویر اشغال می کند مرتبط است. هرچه حیوان دور تر باشد، تشخیص اولیه آن برای مغز سخت تر است.

روش آزمایش: مدل HMAX

با استفاده از کد داده شده، ابتدا داده ها را به دو بخش ترین و تست همانند بخش قبلی تقسیم می کنیم که هر کدام از دسته داده ها نیز خود ۴ کتگوری سر، فاصله نزدیک، فاصله متوسط و فاصله دور دارند. پس از دسته بندی داده ها، ابتدا برای هر کتگوری و سپس برای کل عکس ها فرایند ترین و تست را انجام می دهیم به این صورت که آدرس فولدر ها را برای ترین و تست به برنامه می دهیم و خروجی می گیریم.

با استفاده از قطعه کد زیر در تابع demoRelease.m آدرس دهی انجام می شود.

```
%specify directories for training and testing images
train_set.pos    = 'train/F_a';
train_set.neg    = 'train/F_d';
test_set.pos     = 'test/F_a';
test_set.neg     = 'test/F_d';
```

در تکه کد بالا فولدر داده های کتگوری تصویر دور داده شده است.

همچنین لازم است که در تابع readAllImages پس از خوانش هر عکس، آن را سیاه و سفید کنیم. تکه کد آن به شرح زیر است:

```
cI{i}{j} = double(rgb2gray(imread([dnames{i} '/' c{i}(j).name]))./255;
```

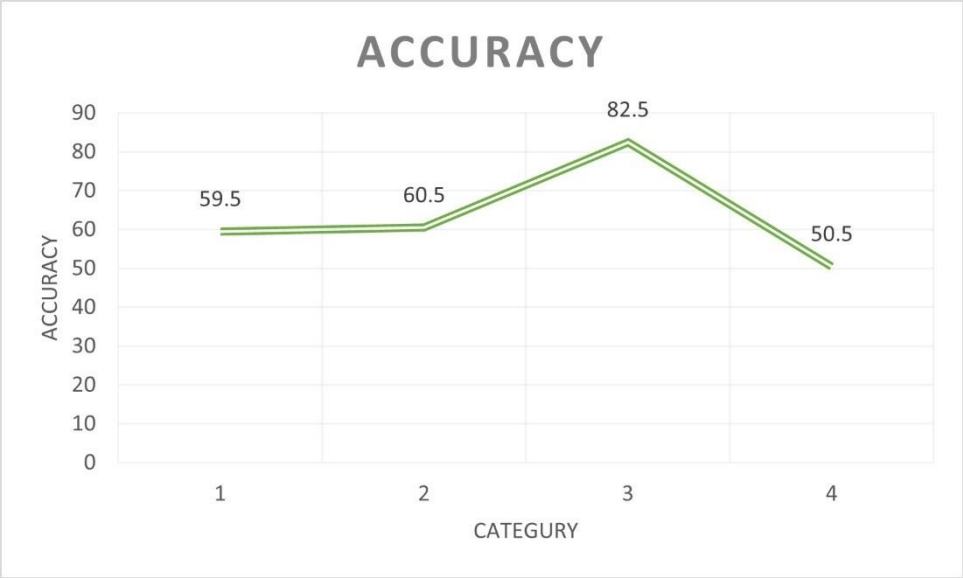
در فرایند طبقه بندی از الگوریتم کمترین فاصله استفاده شده است. که تغییر الگوریتم از SVM به کمترین فاصله با استفاده از تغییر متغیر useSVM از ۱ به ۰ انجام می شود.

با استفاده از مدل و عکس های داده شده، دیتای دقت تشخیص حیوان یا عدم حیوان در این مدل بدست می آید که در قسمت نتایج آورده می شوند.

نتایج: مدل HMAX

عکس ها را در هر کتگوری به صورت جداگانه به مدل می دهیم به صورتی که از هر کتگوری ۳۰۰ عکس موجود است که ۱۰۰ تای آن برای ترینینگ بین حیوان و غیر حیوان به مدل داده می شود و ۲۰۰ تای دیگر نیز برای تست بین حیوان و غیر حیوان استفاده می شود. در آخر نیز ۴۰۰ تا دیتا به صورت مساوی از همه کتگوری های و نوع ها به مدل داده می شود تا به صورت کلی ترین شود، سپس ۸۰۰ تا داده تست به عنوان ورودی داده می شود تا خروجی نهایی مدل بدست آید.

میزان دقت در تشخیص مدل به ازای هر کتگوری به شکل زیر می شود:



دقت کلی مدل در ترین و تست کلیه عکس ها به میزان ۶۸.۵ درصد بوده است.

همانطور که در دیتای بدست آمده از ۴ کتگوری می بینیم، مدل عملکرد بسیار خوبی در تشخیص تصاویر حاوی فاصله متوسط از حیوان داشته است اما در تشخیص تصاویر دور چندان موفق نبوده است. در دو کتگوری اول تقریبا عملکردی یکسان داشته است.

نتایج بدست آمده از مدل مطابقت خوبی با نتایج ذکر شده در مقاله دارند زیرا که در آن جا نیز عملکرد در فاصله متوسط بهترین و عملکرد فاصله دور بدترین بود.

اگر بخواهیم عملکرد کلی که ۶۸.۵ درصد بوده است را نیز بررسی کنیم میبینیم که می توان گفت دقت خوبی بوده است اما می توانست بهتر باشد.

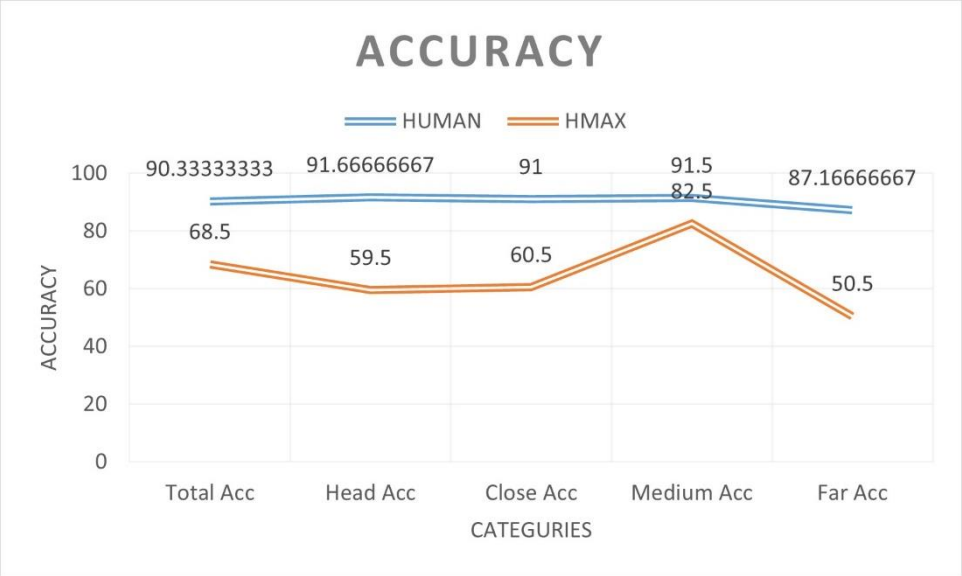
نتیجه گیری

نتایج بدست آمده از دو مرحله تولید شدند که اولین نتایج مربوط به بخش تسک رفتاری آزمایش بودند که در آن افراد باید فرق بین حیوان و غیر حیوان را در طول پخش ۲۰ میلی ثانیه ای محرک متوجه می شدند. اما نتایج بدست آمده در بخش دوم حاصل از عملکرد مدل HMAX در تشخیص حیوان از غیر حیوان پس از فرایند ترینینگ بود.

هدف از آزمایش بررسی عملکرد انسان نسبت به مدل در فرایند دسته بندی حیوان از غیر حیوان بود.

همانطور که دیده می شود، عملکرد انسان از عملکرد مدل بسیار بهتر بوده است. این نتیجه بدست آمده بر خلاف چیزی است که در مقاله بدست آمده بود. نتایج متفاوت بدست آمده می توانند ناشی از ترینینگ کم مدل یا تفاوت پیاده سازی تسک یا مسائل دیگر باشند.

البته همانطور که دیده می شود، هم مدل و هم انسان در تشخیص عکس های کتگوری دور نسبت به کتگوری های نزدیک مشکل داشته اند وو دقت پایین تری را بدست آورده اند.



در نمودار مقایسه ای انسان و مدل می بینیم که عملکرد انسان تقریبا خطی بوده است و در حدود ۹۰ درصد دقت داشته است اما مدل در تصاویر با کتگوری های متفاوت تفاوت فاحشی داشته است و به طور میانگین نیز دقت آن در حدود ۱۲ درصد کمتر از انسان بوده است.

شاید بتوان گفت که تفاوت وجود دارد اما مدل تا حد خوبی توانسته است صرفا در حالت استفاده از مسیر های Feed forward شبیه انسان عمل کند و عملکرد بینایی را مدلسازی کند.

توضیحاتی درباره مدل HMAX

مدل HMAX یک مدل محاسباتی تشخیص اشیا در قشر است که توسط Riesenhuber و Poggio در سال ۱۹۹۹ توسعه یافت. این مدل بر اساس جریان ونترال ویژن است به صورت سلسله مراتبی از نواحی مغز است که تصور می شود واسطه تشخیص اشیاء در قشر مغز هستند. در مورد خواص تغییر ناپذیری و تنظیم شکل نورون ها در قشر inferotemporal macaque، بالاترین ناحیه بینایی در جریان شکمی نشان داده شده است که این مدل نتیجه تجربی انسانی را می تواند پیش‌بینی کند.

مدل HMAX یک مدل سلسله مراتبی است که از چهار سطح تشکیل شده است: S۱، C۱، S۲، و C۲. لایه S۱ لایه ورودی است که داده های تصویر خام را دریافت می کند. لایه C۱ اولین سطح پردازش است که ویژگی های محلی را از تصویر ورودی استخراج می کند. لایه S۲ دومین سطح پردازش است که ویژگی های پیچیده تری را از خروجی لایه C۱ استخراج می کند. لایه C۲ آخرین سطح پردازش است که رده موجود را نشان می دهد.

مدل HMAX یک مدل با انگیزه بیولوژیکی است که سازماندهی چند مرحله اول را برای مسیر بینایی انسان تقلید می کند. این مدل سعی می‌کند مکانیزم صرفا Feed forward را برای تشخیص سریع شی یا موجود که در ۱۰۰-۱۵۰ میلی‌ثانیه اول پردازش بصری رخ می‌دهد را توضیح دهد. این مدل قادر به دستیابی به عملکرد نزدیک به سطح انسانی در چندین کار تشخیص سریع اشیاست.

توابع کد داده شده:

کد داده شده مدل پیاده سازی شده در متلب است که از چند تابع تشکیل شده است. هر کدام از توابع عملکردی را انجام می دهند. به صورت کلی این توابع به دسته های کد اصلی، لایه های مدل، توابع طبقه بندی کننده و ... تقسیم می شوند.

توابع C۱.m و C۲.m همان لایه های S۱ و C۱ و S۲ و C۲ هستند که مدلسازی شده اند.

توابع CLSnn.m و CLSnnC.m برای طبقه بندی کننده با الگوریتم نزدیک ترین همسایه هستند.

توابع CLSosusvm.m و CLSosusvmC.m توابع برای طبقه بند SVM هستند.

تابع readALLImage.m تمامی تصاویر را می خواند و در سلول ذخیره می کند.

تابع padimage.m و unpadimae.m توابع مخصوص اضافه کردن یا برداشتن فریم به دور تصویر هستند.

تابع init_gabor.m نقش فیلتر گابور را بازی می کند.

تابع sumfilter.m نیز نقش جمع کننده محلی را ایفا می کند.

تابع maxfilter.m از تصویر مرحله قبل ماکسیمم گیری می کند. توابع مکس و فیلتر گابور شبیه همان فعالیت سلول های لایه C۱ و S۱ هستند.

نحوه عملکرد مدل HMAX:

مدل Max-pooling سلسله مراتبی (HMAX) یک چارچوب محاسباتی است که برای شبیه سازی پردازش سلسله مراتبی اطلاعات بصری در سیستم بینایی نخستی ها، به ویژه مسیر ونترال طراحی شده است. این مدل توسط پوجیو و همکارانش به عنوان راهی برای ثبت ویژگی های کلیدی مسیرهای پردازش بصری مشاهده شده در مغز پیشنهاد شد.

در اینجا توضیح ساده ای از نحوه عملکرد HMAX آورده شده است:

۱. سلول های ساده و پیچیده:

مدل با لایه ای از سلول های ساده و پیچیده شروع می شود. سلول های ساده به جهت گیری های خاص در یک مکان خاص پاسخ می دهند و میدان های دریافتی موجود در قشر بینایی اولیه را تقلید می کنند. سلول‌های پیچیده، پاسخ‌های سلول‌های ساده را با میدان‌های پذیرای کمی تغییر داده‌اند تا به تغییر ناپذیری ترجمه دست یابند.

۲. مسیر S-C (مسیر ساده-پیچیده):

پاسخ سلول های ساده و پیچیده مسیر ساده-کمپلکس را تشکیل می دهد که نشان دهنده مراحل اولیه پردازش بصری است.

۳. لایه C۱ (لایه S۱):

لایه C۱ که به نام S۱ نیز شناخته می شود، شامل یک عملیات جمع آوری حداکثر بر روی پاسخ سلول های ساده و پیچیده است. این ادغام به دستیابی به درجه ای از ترجمه و عدم تغییر مقیاس کمک می کند. خروجی یک نمایش نمونه پایین از ورودی است که بر فعال‌ترین ویژگی‌ها تمرکز دارد.

۴. لایه S۲:

لایه S۲ ویژگی های پیچیده تری را با اعمال همان اصول سلول های ساده و پیچیده و حداکثر ادغام در خروجی لایه C۱ به تصویر می کشد. این فرآیند به روشی سلسله مراتبی برای ساخت یک سری آشکارسازهای ویژگی پیچیده‌تر تکرار می‌شود.

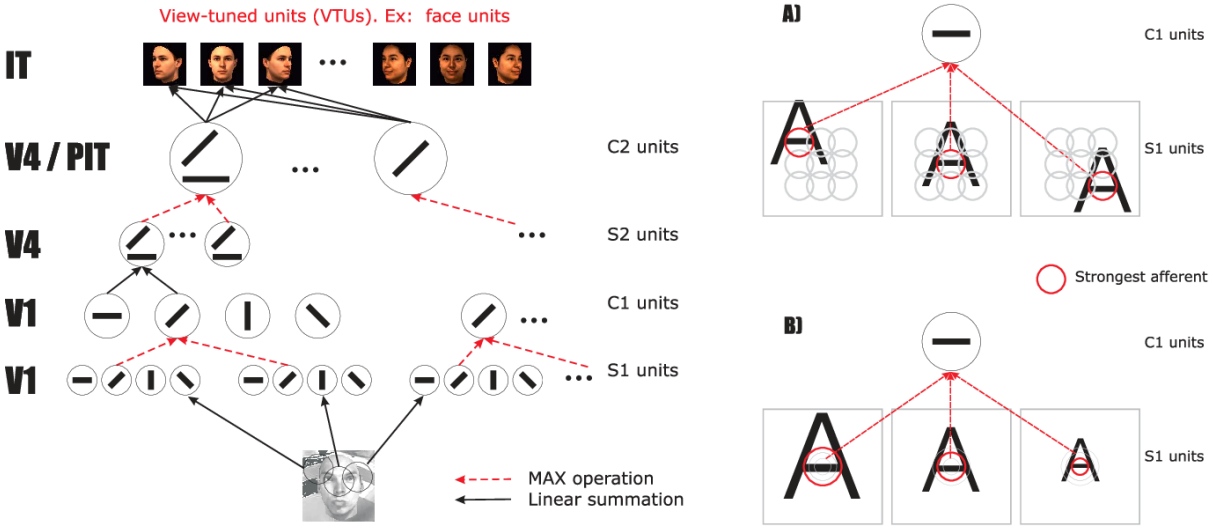
۵. لایه C۲:

لایه C۲ با انجام max-pooling بر روی پاسخ های لایه S۲، تغییر ناپذیری ترجمه و مقیاس را بیشتر افزایش می دهد. این لایه برای دستیابی به تشخیص قوی شی بسیار مهم است.

۶. تشخیص شی:

مرحله نهایی شامل مقایسه خروجی لایه C۲ با قالب های ذخیره شده یا نمونه های اولیه اشیاء است. مدل اشیاء را بر اساس بالاترین امتیازهای تطبیق شناسایی می کند و نتیجه یک پاسخ تشخیص است.

سازماندهی سلسله مراتبی و ترکیب عملیات max-pooling در مراحل مختلف به HMAX اجازه می دهد تا ویژگی های اساسی سیستم بینایی نخستی ها را به تصویر بکشد. این مدل در درک تشخیص اشیا در مغز تأثیرگذار بوده است و کاربردهایی در وظایف بینایی رایانه پیدا کرده است که به عنوان پلی بین علوم اعصاب و هوش مصنوعی عمل می کند.



الگوریتم KNN:

الگوریتم K-Nearest Neighbors (KNN) یک روش یادگیری ماشینی ساده و در عین حال قدرتمند است که برای مسائل طبقه‌بندی و رگرسیون استفاده می‌شود. این یک الگوریتم ناپارامتریک است، به این معنی که هیچ فرضی در مورد توزیع داده های اساسی ایجاد نمی کند. الگوریتم بر اساس مفهوم شباهت است، جایی که یک نقطه داده جدید بر اساس شباهت آن به داده های آموزشی طبقه بندی می شود. این الگوریتم K-Nearest Neighbors نامیده می شود زیرا K نزدیکترین همسایه را به نقطه داده جدید در مجموعه داده آموزشی در نظر می گیرد.

الگوریتم KNN یک الگوریتم یادگیری نظارت شده است که به طور گسترده در تشخیص الگو، داده کاوی و تشخیص نفوذ استفاده می شود. این یک الگوریتم کاربر پسند است که به راحتی قابل درک و پیاده سازی است. این الگوریتم مبتنی بر این ایده است که اشیاء مشابه احتمالاً متعلق به یک کلاس هستند یا مقادیر مشابهی دارند همچنین با محاسبه فاصله بین نقطه داده جدید و سایر نقاط داده در مجموعه داده آموزشی کار می کند. سپس K نزدیکترین همسایه ها انتخاب می شوند و کلاس یا مقدار نقطه داده جدید با رأی اکثریت یا میانگین همسایگان K تعیین می شود.

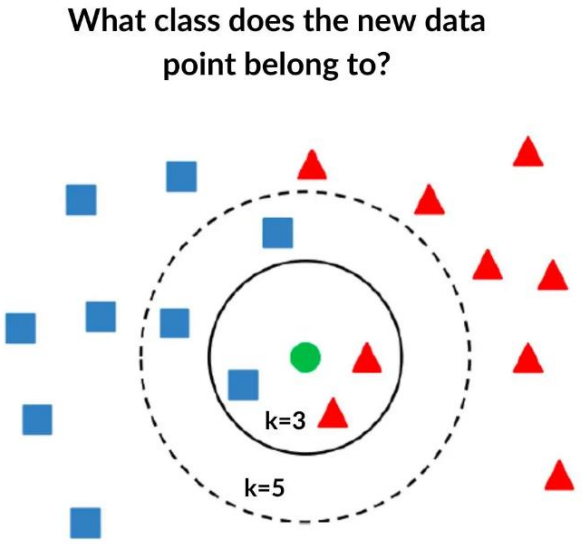
الگوریتم KNN یک الگوریتم همه کاره است که می تواند داده های عددی و دسته بندی را مدیریت کند. نسبت به الگوریتم های دیگر حساسیت کمتری دارد. این الگوریتم همچنین قادر به مدیریت داده های از دست رفته با نسبت دادن مقادیر از دست رفته با میانگین یا میانه داده های موجود است. از نظر محاسباتی برای مجموعه داده های کوچک کارآمد است اما برای مجموعه داده های بزرگ می تواند کند باشد.

الگوریتم KNN چندین پارامتر دارد که برای دستیابی به عملکرد بهینه باید تنظیم شوند. مهمترین پارامتر K است که تعیین کننده تعداد همسایگانی است که باید در نظر گرفته شوند. مقدار کم K می تواند منجر به بیش از حد برازش شود، در حالی که مقدار زیاد K می تواند منجر به عدم تناسب شود. مقدار K را می توان با استفاده از تکنیک های اعتبار سنجی متقابل تعیین کرد.

این الگوریتم چندین معیار فاصله دارد که می توان از آنها برای محاسبه فاصله بین نقاط داده استفاده کرد. متداول ترین متریک فاصله، فاصله اقلیدسی است که فاصله خط مستقیم بین دو نقطه در فضای اقلیدسی است. سایر معیارهای فاصله شامل فاصله منهتن، فاصله مینکوفسکی و فاصله همینگ هستند.

در زمینه های مختلف از جمله تشخیص تصویر، تشخیص گفتار و سیستم های توصیه کاربردهای متعددی دارد. در تشخیص تصویر، از این الگوریتم برای طبقه بندی تصاویر بر اساس شباهت آنها با سایر تصاویر در مجموعه داده آموزشی استفاده می شود. در تشخیص گفتار، این الگوریتم برای تشخیص کلمات گفتاری بر اساس شباهت آنها به سایر کلمات در مجموعه داده آموزشی استفاده می شود. در سیستم های توصیه، الگوریتم برای توصیه محصولات یا خدمات بر اساس شباهت آنها به سایر محصولات یا خدمات در مجموعه داده آموزشی استفاده می شود.

به طور خلاصه، الگوریتم K-Nearest Neighbors (KNN) یک روش یادگیری ماشینی ساده و در عین حال قدرتمند است که برای مسائل طبقه‌بندی و رگرسیون استفاده می‌شود. الگوریتم مبتنی بر مفهوم شباهت است و K نزدیکترین همسایه به نقطه داده جدید در مجموعه داده آموزشی را در نظر می گیرد. این الگوریتم همه کاره است و می تواند داده های عددی و مقوله ای را مدیریت کند. الگوریتم چندین پارامتر دارد که برای دستیابی به عملکرد بهینه باید تنظیم شوند. این الگوریتم چندین متریک فاصله دارد که می توان از آنها برای محاسبه فاصله بین نقاط داده استفاده کرد. این الگوریتم کاربردهای متعددی در زمینه های مختلف از جمله تشخیص تصویر، تشخیص گفتار و سیستم های توصیه دارد.



الگوریتم SVM:

الگوریتم ماشین بردار پشتیبان (SVM) یک الگوریتم یادگیری ماشینی قدرتمند است که برای کارهای طبقه‌بندی و رگرسیون استفاده می‌شود. این یک الگوریتم یادگیری نظارت شده است که می تواند برای داده های خطی و غیرخطی استفاده شود. الگوریتم SVM با یافتن ابر صفحه بهینه کار می کند که نقاط داده را به کلاس های مختلف جدا می کند. ابرصفحه طوری انتخاب می شود که حاشیه بین نزدیکترین نقاط طبقات مختلف به حداکثر برسد. الگوریتم SVM به طور گسترده در زمینه های مختلف از جمله تشخیص تصویر، طبقه بندی متن و بیوانفورماتیک استفاده می شود.

الگوریتم SVM مبتنی بر مفهوم مرز تصمیم است که یک خط یا صفحه ای است که نقاط داده را به کلاس های مختلف جدا می کند. الگوریتم SVM سعی می کند مرز تصمیم گیری را پیدا کند که حاشیه بین نزدیک ترین نقاط کلاس های مختلف را به حداکثر می رساند. حاشیه فاصله بین مرز تصمیم و نزدیکترین نقاط طبقات مختلف است. الگوریتم SVM طبقه بندی کننده حاشیه حداکثر نامیده می شود زیرا سعی می کند مرز تصمیم را با حداکثر حاشیه پیدا کند.

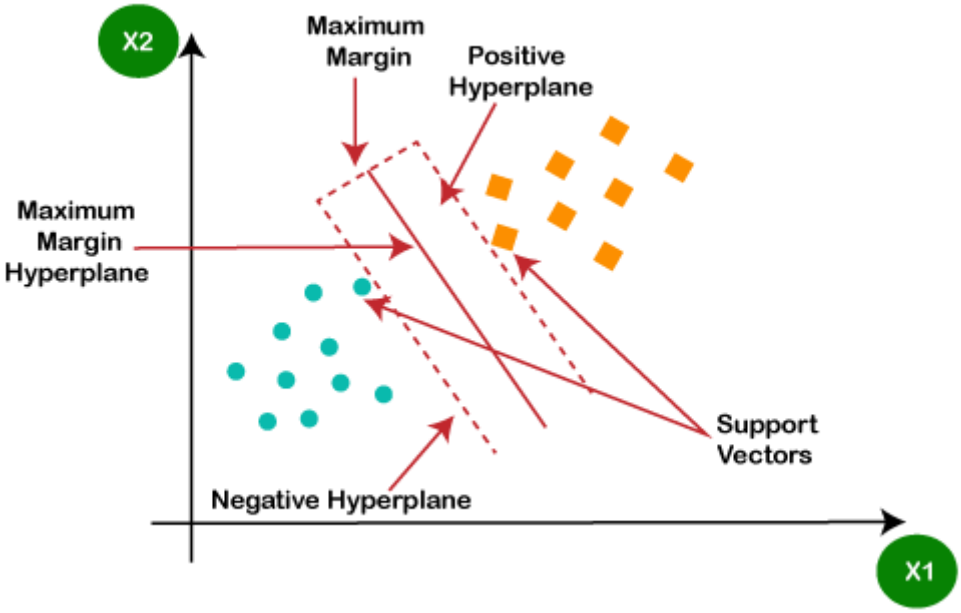
این الگوریتم می تواند داده های خطی و غیرخطی را مدیریت کند. برای داده های خطی، الگوریتم SVM از یک مرز تصمیم گیری خطی استفاده می کند ولی برای داده های غیرخطی، الگوریتم SVM از یک مرز تصمیم غیرخطی استفاده می کند. همچنین از یک تابع هسته برای تبدیل داده ها به فضایی با ابعاد بالاتر استفاده می کند، جایی که می توان از یک مرز تصمیم گیری خطی برای جداسازی نقاط داده استفاده کرد.

SVM دارای چندین مزیت نسبت به سایر الگوریتم های یادگیری ماشینی است. در فضاهای با ابعاد بالا، جایی که تعداد ویژگی ها بسیار بیشتر از تعداد نقاط داده است، موثر است. همچنین در مواردی که تعداد ویژگی ها از تعداد نقاط داده بیشتر باشد نیز می تواند موثر باشد. الگوریتم SVM در مقایسه با الگوریتم های دیگر کمتر مستعد بیش از حد برازش است. همچنین از نظر محاسباتی برای مجموعه داده های کوچک تا متوسط کارآمد است.

چندین پارامتر دارد که برای دستیابی به عملکرد بهینه باید تنظیم شوند. مهمترین پارامتر تابع هسته است که شکل مرز تصمیم را تعیین می کند. پارامترهای دیگر شامل پارامتر تنظیم، که مبادله بین حاشیه و نرخ طبقه‌بندی اشتباه را کنترل می‌کند، و پارامتر گاما، که شکل تابع هسته را کنترل می‌کند.

الگوریتم SVM کاربردهای متعددی در زمینه های مختلف دارد. در تشخیص تصویر، برای طبقه بندی تصاویر بر اساس ویژگی های آنها استفاده می شود. در طبقه بندی متن، برای طبقه بندی اسناد متنی بر اساس محتوای آنها استفاده می شود. در بیوانفورماتیک، برای طبقه بندی ژن ها بر اساس سطح بیان آنها استفاده می شود.

به طور خلاصه، الگوریتم SVM یک الگوریتم یادگیری ماشینی قدرتمند است که برای کارهای طبقه‌بندی و رگرسیون استفاده می‌شود. این الگوریتم مبتنی بر مفهوم مرز تصمیم است و سعی می کند ابر صفحه بهینه را که نقاط داده را به کلاس های مختلف جدا می کند، بیابد. الگوریتم SVM می تواند داده های خطی و غیرخطی را مدیریت کند و در فضاهای با ابعاد بالا موثر است. الگوریتم چندین پارامتر دارد که برای دستیابی به عملکرد بهینه باید تنظیم شوند. الگوریتم SVM در زمینه های مختلف از جمله تشخیص تصویر، طبقه بندی متن و بیوانفورماتیک کاربردهای متعددی دارد.



مراجع

Supplementary Web Material: Serre et al, PNAS 2007 (mit.edu)

Ghodrati-Report on Hmax

Serre, Thomas, Aude Oliva, and Tomaso Poggio. "A feedforward architecture accounts for rapid categorization." *Proceedings of the national academy of sciences* 104.15 (2007): 6424-6429.

Folowosele, Fopefolu, R. Jacob Vogelstein, and Ralph Etienne-Cummings. "Towards a cortical prosthesis: Implementing a spike-based HMAX model of visual object recognition in silico." *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 1.4 (2011): 516-525.