UNIVERSIDADE FEDERAL RURAL DO RIO DE JANEIRO
INSTITUTO MULTIDISCIPLINAR

MATHEUS OLIVEIRA DE BARROS
NELSON ANTUNES LARANJEIRA NETO

Classificação Automática de Anomalias em Vídeos

Prof. Filipe Braida do Carmo, D.Sc.
Orientador

Nova Iguaçu, Dezembro de 2020

Classificação Automática de Anomalias em Vídeos

Matheus Oliveira de Barros

Nelson Antunes Laranjeira Neto

Projeto Final de Curso submetido ao Departamento de Ciência da Computação do Instituto de Matemática da Universidade Federal Rural do Rio de Janeiro como parte dos requisitos necessários para obtenção do grau de Bacharel em Informática.

Apresentado por:

_____

Matheus Oliveira de Barros

_____

Nelson Antunes Laranjeira Neto

Aprovado por:

_____

Prof. Filipe Braida do Carmo, D.Sc.

_____

Prof. Leandro Guimaraes Marques Alvim, D.Sc.

_____

Prof. Bruno José Dembogurski, D.Sc.

NOVA IGUAÇU, RJ - BRASIL

Dezembro de 2020

Classificação Automática de Anomalias em Vídeos

Matheus Oliveira de Barros

Nelson Antunes Laranjeira Neto

Projeto Final de Curso submetido ao Departamento de Ciência da Computação do Instituto de Matemática da Universidade Federal Rural do Rio de Janeiro como parte dos requisitos necessários para obtenção do grau de Bacharel em Informática.

Apresentado por:

_____

Matheus Oliveira de Barros

_____

Nelson Antunes Laranjeira Neto

Aprovado por:

_____

Prof. Filipe Braida do Carmo, D.Sc.

_____

Prof. Leandro Guimaraes Marques Alvim, D.Sc.

_____

Prof. Bruno José Dembogurski, D.Sc.

NOVA IGUAÇU, RJ - BRASIL

Dezembro de 2020

Emitido em 08/12/2020

DOCUMENTOS COMPROBATÓRIOS Nº 12924/2020 - CoordCGCC (12.28.01.00.00.98)

(Nº do Protocolo: NÃO PROTOCOLADO)

(Assinado digitalmente em 15/12/2020 15:48)
BRUNO JOSE DEMBOGURSKI
PROFESSOR DO MAGISTERIO SUPERIOR
DeptCC/IM (12.28.01.00.00.83)
Matrícula: 2124964

(Assinado digitalmente em 16/12/2020 15:13)
FILIPE BRAIDA DO CARMO
PROFESSOR DO MAGISTERIO SUPERIOR
DeptCC/IM (12.28.01.00.00.83)
Matrícula: 3929524

(Assinado digitalmente em 15/12/2020 14:19)
LEANDRO GUIMARAES MARQUES ALVIM
PROFESSOR DO MAGISTERIO SUPERIOR
DeptCC/IM (12.28.01.00.00.83)
Matrícula: 1800852

(Assinado digitalmente em 15/12/2020 17:39)
NELSON ANTUNES LARANJEIRA NETO
DISCENTE
Matrícula: 2015780317

(Assinado digitalmente em 15/12/2020 13:26)
MATHEUS OLIVEIRA DE BARROS
DISCENTE
Matrícula: 2015780287

Para verificar a autenticidade deste documento entre em https://sipac.ufrrj.br/documentos/ informando seu número: **12924**, ano: **2020**, tipo: **DOCUMENTOS COMPROBATÓRIOS**, data de emissão: **15/12/2020** e o código de verificação: **4f1920125e**

Agradecimentos

Matheus Oliveira de Barros

Primeiramente agradeço a minha família que nunca deixou de me apoiar e não permitiu que em nenhum momento eu perdesse a força para continuar em busca do meu objetivo.

Agradeço aos meus amigos por compartilhar ótimos momentos comigo e sempre fornecer ajuda para que eu conseguisse continuar seguindo em frente. Em especial, quero agradecer aos meus amigos Ryan e Nelson por todo o apoio e dedicação, possibilitando com que os momentos mais desafiadores fossem superados com bom humor e "gostosas gargalhadas".

Agradeço aos amigos do grupo "Leev 2 eet"pelos vários momentos descontraídos, conversas, brincadeiras e ensinamentos. Sem o apoio deles eu não estaria onde estou. Parte das minhas conquistas foram possíveis por causa deles, então, muito obrigado.

Por fim, tenho muito a agradecer aos professores pelos conselhos e direcionamentos passados ao longo do curso, devido a isso fui capaz de desenvolver tanto minhas habilidades pessoais quanto profissionais.

Nelson Antunes Laranjeira Neto

Gostaria de agradecer primeiramente à minha mãe, ao meu pai e ao meu irmão por estarem comigo do começo ao fim dessa jornada e por todo o apoio e amor. Agradeço também a todos os meus amigos que me incentivaram e sempre estiveram lá quando precisei. Eu não conseguiria chegar aqui sem o suporte de todos. Obrigado.

A todo mundo do Leev2Eet por todos os momentos de amizade, risadas e aprendizado que passamos juntos, desde o primeiro período de faculdade. Agradeço em especial aos meus melhores amigos: Ryan – uma pessoa incrivelmente estranha que sei que sempre vou poder contar em qualquer situação por mais bizarra que seja; Pararati – que sempre me ajudou a me manter confiante, a superar os desafios e por sempre ser um grande líder; Vitor – por me ajudar a manter a calma, por todos os ensinamentos e por sempre aparecer nos churrascos; Brinquedo – por me convencer a comprar jogos horríveis na Steam; e Esdras – por ser alguém com quem eu sempre possa conversar sobre as coisas mais profundas da vida.

Agradeço também ao meu orientador Filipe Braida por ter toda a paciência do mundo durante a elaboração desse texto, por todas as dicas que vêm desde o primeiro período de faculdade e por confiar no meu potencial. Também sou grato a todos os professores pelo conhecimento e por me ajudar a me tornar quem eu sou hoje.

RESUMO

Classificação Automática de Anomalias em Vídeos

Matheus Oliveira de Barros e Nelson Antunes Laranjeira Neto

Dezembro/2020

Orientador: Filipe Braida do Carmo, D.Sc.

A quantidade de câmeras de vigilância tem aumentando significativamente nos últimos anos com o objetivo de observar a conduta das pessoas a fim de evitar incidentes indesejados ou prestar algum tipo de socorro imediato. Devido a esse aumento tornou-se expressivamente mais difícil a tarefa de observar e avaliar esses incidentes em uma quantidade imensa de câmeras. A detecção de anomalias em vídeos vem se tornando uma ferramenta indispensável no setor de segurança para o monitoramento de diferentes áreas através de grandes quantidades de câmeras de vigilância, mas para cada câmera, normalmente, é necessário calcular um limiar para separar os eventos anômalos. Nosso trabalho propõe uma forma de detecção de anomalias através da classificação da reconstrução de frames gerados por um Autoencoder Convolucional.

ABSTRACT

Classificação Automática de Anomalias em Vídeos

Matheus Oliveira de Barros and Nelson Antunes Laranjeira Neto

Dezembro/2020

Advisor: Filipe Braida do Carmo, D.Sc.

The number of surveillance cameras has increasing significantly in the last years with the aim of observer the peoples's conduct in order to avoid unwanted incidents or to provide some kind of immediate help and, due to this increase, the task of observing and to evaluating these incidents in a large number of surveillance cameras have become expressively more difficult. Anomaly detection in videos is becoming an indispensable tool to monitoring large amounts of surveillance cameras, however, each camera must calculate a threshold to separate anomalous events. Our research proposes a way to detect anomalies through the classification of the reconstrution of frames created by a Convolutional Autoencoder

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

A análise de vídeo vem ganhando importância na área de reconhecimento de padrões, tendo como algumas tarefas possíveis: classificação de dados, reconhecimento de faces, reconhecimento de voz e outras. Nos últimos anos, a busca por soluções envolvendo identificação de eventos anômalos vêm ganhando bastante relevância devido ao crescimento do meio digital envolvendo monitoramento de vídeo considerando diferentes ambientes. Assim, faz-se necessário o uso de formas eficientes para análise das diversas situações presentes nos vídeos, de modo que seja possível detectar ocorrências de anomalias.

Uma anomalia, ou também chamada de anormalidade, pode ser definida como qualquer evento incomum aqueles que frequentemente ocorrem em um determinado local. Ao considerar anomalias em vídeo, esta é definida como um objeto ou comportamento identificado nas imagens que é diferente do que é considerado normal.

A ocorrência de anomalias pode ser identificada em diferentes contextos. Quando a ocorrência é simples, por exemplo, um grande assalto a um banco, é trivial para uma pessoa conseguir identificar esse evento, porém ao considerar situações mais complexas como a presença de um pedestre em local restrito em meio a multidão, fica mais difícil utilizar a análise humana.

Um dos problemas com a detecção manual de anomalias em vídeo é o tempo investido para identificá-las. É necessário uma longa análise do vídeo para que consiga

entender todos os detalhes presentes na cena para então julgar se há anomalia ou não. Além disso, também ocorre o problema de múltiplas interpretações para o mesmo caso.

A necessidade de grande tempo para detecção manual de anomalias em vídeo e a possível perda de detalhes são fatores importantes que precisam ser levados em consideração. Não considerar todos os detalhes faz com que o processo de tomada de decisão em relação a detecção de anomalias seja afetado, podendo ocasionar em uma decisão errada.

Tanto o alto tempo para análise quanto a perda de detalhes torna inviável o processo de detecção de anomalias de forma manual ao aplicado em áreas críticas onde o erro deve sempre ser minimizado, como por exemplo o setor de segurança. Neste caso, todo o processo se torna ineficiente quando feito manualmente, podendo ocasionar em uma decisão equivocada.

No caso do monitoramento de vídeos, considerando que há diversas câmeras com imagens de diferentes contextos, é fundamental o uso de métodos automáticos que consigam identificar anomalias em um curto período de tempo. Além disso, os métodos precisam ser confiáveis e seguros, evitando a resolução errada para uma possível anomalia.

Com base no problema de detecção de anomalias, muitos pesquisadores desenvolveram projetos capazes de identificar anomalias rapidamente utilizando inteligência artificial. Na literatura, é possível encontrar diversos trabalhos, tanto para um contexto específico quanto para contexto mais genérico. Devido a grande variedade de situações possíveis de ocorrência de anomalia, há diferentes estratégias de abordagem para esse problema utilizando ou não inteligência artificial.

Esse trabalho busca desenvolver uma técnica de detecção de anomalias em vídeo utilizando ferramentas de aprendizado supervisionado e não supervisionado. Através desta técnica será possível reduzir consideravelmente o tempo de análise e a perda de detalhes para identificar cada ocorrência de anomalia, uma vez que esta tarefa estará sendo realizada por um processo automático que possui a habilidade de aprender as

características fundamentais do ambiente presente no vídeo de maneira autônoma.

A técnica desenvolvida neste trabalho utiliza para aprender as principais características dos objetos presentes nos vídeos um modelo Autoencoder Convolucional. Com o uso desse modelo será possível aplicar a técnica em diferentes ambientes devido a sua capacidade de compreender e reproduzir imagens mantendo o erro baixo quando comparado à imagem original.

1.1 Organização

Este trabalho está organizado da seguinte maneira:

O capítulo 2 apresenta todos os conceitos fundamentais para o entendimento das ferramentas utilizadas na elaboração da técnica de detecção de anomalias presente neste trabalho

No capítulo 3 será apresentada a técnica proposta para detecção de anomalias, bem como a objetivo de cada modelo utilizado.

O capítulo 4 possui a metodologia utilizada para a experimentação, assim como a base de dados utilizada e a análise do resultado obtido.

Por fim, é apresentada a conclusão e as referências bibliográficas utilizadas para o desenvolvimento deste trabalho.

Capítulo 2

Fundamentação Teórica

A fim de possibilitar a total compreensão das técnicas e tecnologias utilizadas, este capítulo apresenta os conceitos fundamentais que permitiram o desenvolvimento de uma técnica para detecção de anomalias em vídeo. As seções abordam os seguintes assuntos: detecção de anomalias, onde é definido de fato o que é uma anomalia e suas características, bem como as dificuldades para detecção; Aprendizado de Máquina, nesta seção será detalhado um conjunto de técnicas utilizadas neste trabalho.

2.1 Detecção de anomalia

O reconhecimento de padrões é uma área da ciência que tem como objetivo encontrar uma classificação possível para diferentes objetos baseando-se em categorias ou classes. Tais objetos são conhecidos como objetos de estudo e podem variar de acordo com a aplicação. Essas aplicações são distribuídas em diferentes áreas devido ao grande ganho de conhecimento obtido pela identificação de características ou comportamentos. (THEODORIDIS; KOUTROUMBAS, 2006)

A importância do reconhecimento de padrões surgiu com a automação da produção em diferentes setores, sendo necessária a elaboração de mecanismos que fossem capazes de reproduzir as tarefas de forma muito mais eficiente em relação ao tempo e qualidade. A aplicação desta área se tornou tão abrangente que nos tempos atuais

tornou-se uma das partes fundamentais da maioria dos sistemas de tomada de decisão. (THEODORIDIS; KOUTROUMBAS, 2006)

Além do reconhecimento de padrões, o momento em que um comportamento inesperado interrompe o que outrora era considerado padrão, mesmo que por alguns instantes, possui grande relevância. Tal relevância ocorre devido aos possíveis impactos que este pode proporcionar no contexto em que foi identificado. Para esses casos, é dado o nome anomalia.

A ocorrência de anomalias pode ocasionar diferentes impactos no ambiente em que está inserida. Esses impactos variam em nível de importância, pois há situações onde a incidência de anomalia ocorre por alguns instantes, podendo indicar uma falha no sistema, onde esta, por sua vez, pode ser resolvida rapidamente. (THEODORIDIS; KOUTROUMBAS, 2006)

O conceito de anomalia é bastante amplo, permitindo com que seja possível a identificação em diferentes áreas, e levando em consideração o impacto causado por elas, é necessário a existência de métodos que possibilitem a detecção de anomalias em diferentes situações.

Dentre as inúmeras possibilidades de aplicação de detecção de anomalias, serão apresentados alguns exemplos de forma a conceder um entendimento com clareza acerca da ocorrência de anomalias e algumas possíveis formas de incidência.

2.1.1 Aplicações para detecção de anomalias

A detecção de anomalias é frequentemente encontrada em estudos seja para descobrir a ocorrência, tentar encontrar uma explicação para sua incidência ou até mesmo obter uma forma de conseguir prever possíveis casos de anomalias em um determinado ambiente.

O uso de imagens como entrada de dados para detecção de anomalias faz com que seja possível abordar diferentes situações incluindo as do cotidiano. A Figura 2.1 representa a identificação de comportamentos anômalos baseados na distância em que os indivíduos estão uns dos outros, sendo considerado anômalo qualquer comportamento

que indique grande proximidade entre as pessoas.

O exemplo da Figura 2.1, além de poder ser aplicado a diversas circunstâncias comuns, também torna-se relevante em situações críticas e com alto grau de importância, como a análise do comportamento de indivíduos durante uma pandemia de uma doença contagiosa, onde o contato físico precisa ser evitado ao máximo para que a propagação sejam controlada e até mesmo reduzida.

Figura 2.1: Detecção de anomalia baseada na proximidade das pessoas.

Outro exemplo para a utilização de detecção de anomalias em imagens é a ocorrência de eventos causados por algum tipo de comportamento inesperado no tráfego de diferentes rodovias. Esses eventos podem ser de diversos tipos, como: acidente de trânsito, mau funcionamento de um veículo em uma estrada, travessia inadequada de pedestres em uma rodovia de alta velocidade, entre outros.

A Figura 2.2 indica um cenário para detecção de anomalias no contexto de tráfego rodoviário, onde nesse caso a anomalia é compreendida como acidentes ocasionados por um ou mais veículos. As colunas (a) e (c) são situações consideradas normais e esperadas no dia a dia nas rodovias, enquanto que as colunas (b) e (d), apresentam a detecção de acidentes envolvendo os veículos que estavam presentes naquele momento.

Além do uso de imagens para detecção de anomalias, é comum o uso de fontes

Figura 2.2: Neste cenário é possível observar situações consideradas normais e anomalias, onde estas, por sua vez, são caracterizadas pelo choque entre dois ou mais veículos.(Yun et al., 2014)

em outras formas como dados referentes a algum serviço, por exemplo: padrão de tráfego incomum em uma rede de computadores, indicando uma possível invasão ou ataque Kumar (2005), ou elementos incomuns encontrados em dados de ressonância magnética que podem indicar a presença de tumores Spence, Parra e Sajda (2001).

Em todos os contextos onde a detecção de anomalias pode ser aplicada é possível observar os impactos negativos causados por elas. Tais impactos podem variar de acordo com o contexto abordado e o tipo de anomalia, uma vez que alguns tipos podem ser apenas uma mudança de comportamento na região, sendo nesse caso uma situação especial que deve ser levada em consideração nos critérios que definem uma anomalia para o ambiente em questão.

Levando em consideração as consequências provocadas por anomalias, é necessário o uso de algumas técnicas capazes de identificar comportamentos classificados como incomuns. É interessante que essas técnicas sejam eficientes em relação a confiabilidade e tempo, fazendo com o que o dano causado pelas anomalias seja

amenizado e as medidas para correção possam ser tomadas o mais rápido possível.

Sabendo que a detecção de anomalias pode variar tanto em grau de gravidade quanto no tipo, faz-se necessário que tenham diferentes métodos para encontrar a melhor forma de identificá-las. Além disso, é preciso considerar as dificuldades encontradas em um determinado ambiente e o histórico de possíveis mudanças comportamentais para que o método consiga se adaptar.

A seguir serão abordados alguns métodos utilizados na literatura para detecção de anomalias, onde cada um possui suas particularidades e justificativas para aplicação considerando os problemas mencionados anteriormente.

2.1.2 Métodos para detecção de anomalias

Apesar dos impactos negativos causados pela ocorrência de anomalias, é possível também obter informações capazes de promover algum resultado positivo considerando o ambiente ao qual foi encontrada. Na área de negócios, por exemplo, observar pequenas mudanças em determinados cenários torna possível que uma empresa consiga obter grandes lucros devido a rápida tomada de decisão proveniente da detecção desses comportamentos.

Por mais que existam diversas técnicas de detecção de anomalias na literatura, todas enfrentam algumas dificuldades que tornam todo o processo muito mais desafiador do que é em sua teoria. Parte dessa dificuldade surge com base nas diversas possibilidades de ocorrências e cenários distintos.

Um grande problema acerca de detecção de anomalias é a identificação dos comportamentos considerados normais no ambiente em análise. Considerando que há vários fatores que influenciam em um comportamento, tentar obter um padrão pode ser bastante desafiador. Além disso, também é necessário identificar um limite bem definido para afirmar o que é anomalia, de modo que o erro de escolha seja mínimo.

As anomalias podem ocorrer tanto por ações ilegais quanto por ações mais simples quando não há necessariamente a intenção de violar alguma regra ou lei. No geral, nesse caso, as anomalias não são tão críticas e o dano não envolve muitas partes.

Anomalias causadas por ações ilegais costumam ser mais complexas de serem detectadas, uma vez que o elemento causador busca realizar todos os cuidados para que a ação causada seja o mais idêntica possível a que é considerada normal. Logo, saber o que de fato é verdadeiramente normal em um ambiente propício a ações ilegais é uma tarefa desafiadora.

O conceito de anomalia também possui relação ao contexto e época ao qual se encontra, portanto, é possível que o comportamento que hoje é corriqueiro, no passado era considerado intolerado. O mesmo ocorre considerando o futuro, as ações e decisões são dinâmicas e podem variar com o passar do tempo, fazendo com que comportamentos diferentes surjam, mas que não serão considerados irregulares.

Um exemplo simples dessa mudança de comportamento é a alteração da orientação de rodovias, passando de mão única para mão dupla. Nesse caso, a causa da mudança pode ser alguma regulamentação nova posta em vigor após alguns anos após a criação da rodovia que não deve ser considerada uma anomalia, pois a partir do momento da mudança será considerado o novo normal.

Não há uma medida exata que indique o que é e o que não é anomalia, essa definição varia de acordo com o contexto, fazendo com que algumas variações possam ser aceitas em uma determinada aplicação, mas proibida em outras. Por exemplo, variações no preço de um determinado produto são aceitáveis, porém considerando o contexto médico, variações de temperatura humana podem significar alguma anomalia e a causa precisa ser investigada.

Os dados para análise de anomalia possuem ruídos que podem ser entendidos como anomalias, tornando difícil o processo de distinção do que é anomalia e o que é ruído. A existência de ruído pode e não deve ser levado em consideração, uma vez que o ruído pode induzir ao erro, diminuindo a precisão da detecção de anomalias. (RANI; RAO, 2019)

Levando em consideração as dificuldades listadas anteriormente, tornou-se necessário o uso de ferramentas inteligentes para identificação de anomalias em diferentes contextos. Além disso, é necessário também que essas ferramentas sejam automati-

zadas para que seja possível abranger diferentes tipos de situações. Contudo, alguns conceitos da computação que permitem desenvolver aplicações capazes de detectar anomalias através do aprendizado precisam ser introduzidos.

2.2 Aprendizado de Máquina

Aprendizado de máquina é um subcampo da Inteligência Artificial que promove a elaboração de técnicas que possibilitam com que sistemas aprendam a identificar padrões através de análise de dados e tomar decisões com o mínimo de intervenção humana. Sendo assim, a aplicabilidade deste conceito torna-se bastante viável em qualquer tarefa que necessite ser automatizada. (MITCHELL, 1997)

Dentre as diversas áreas em que o Aprendizado de Máquinas pode ser aplicado, tem-se como exemplo os mecanismos de pesquisas que aprendem a atingir melhores resultados em uma busca, software de anti-spam que filtram os e-mails e detecção de fraudes em transações de cartão de crédito. Além disso há também aplicações bem mais complexas como detecção de faces através da câmera dos smartphones e reconhecimento de voz.

Aprendizado de máquina tornou-se bastante relevante após o grande desenvolvimento da tecnologia no mundo, fazendo com que muitos dados fossem gerados. Dessa forma, passou a ser inviável realizar análises manuais para elaboração do processo de tomada de decisão, sendo necessário então criar técnicas capazes de identificar certos padrões nos dados e fornecer algum tipo de informação.

Nos últimos anos o acesso à internet e as novas tecnologias foram ficando cada vez mais populares, tendo como resultado a geração de uma gigantesca quantidade de dados para serem processados. Cada pessoa é capaz de gerar milhões de dados, onde estes, podem ser usados para diferentes propósitos, desde a localização de novos serviços nas redondezas, até o grau de satisfação de um determinado produto.

A principal questão acerca do uso dos dados está em buscar a melhor forma de como interpretá-lo a fim de que seja possível obter um efeito positivo no contexto em que o qual está relacionado. Além disso, também é necessário que o estudo sobre

cada dado seja otimizado para que este possa ser convertido em informação e ser utilizado no momento necessário para ser transformado em conhecimento.

A aquisição de conhecimento é um processo que consiste em dois fatores: aprendizado e experiência. Logo, qualquer pessoa que deseja obter conhecimento sobre um assunto é necessário que haja um estudo sobre o mesmo e experiências que comprovem o que foi aprendido.

Tendo em mente a importância do aprendizado automatizado e as inúmeras situações onde este conceito pode ser aplicado, passa a ser necessário o uso de diferentes tipos de abordagens para que seja possível obter melhores resultados através de variadas técnicas. Com isso, o campo do Aprendizado de Máquina precisou ser ramificado em subcampos capazes de realizar tarefas específicas que variam de acordo com a necessidade.

2.2.1 Tipos de Aprendizado de Máquina

O aprendizado é um conceito bastante amplo e claramente aplicado em diferentes formas no cotidiano. Essa variedade se faz necessária devido ao dinamismo presente em cada situação. O aprendizado na computação, quando aplicado em circunstâncias reais e complexas, necessita de paradigmas distintos para ser utilizado com maior eficiência. (SHALEV-SHWARTZ; BEN-DAVID, 2014)

Os principais tipos de Aprendizado de Máquina são: aprendizado supervisionado, aprendizado não-supervisionado, aprendizado semi-supervisionado e aprendizado por reforço. Cada tipo de aprendizado possui características específicas que permite com que a aplicação do conhecimento na computação seja mais abrangente, promovendo diferentes situações onde o aprendizado possa ser utilizado.

De acordo com Braida do Carmo (2012), o aprendizado supervisionado tem como objetivo utilizar exemplos de dados com suas respostas exatas de modo que possibilite a generalização para outros tipos de dados. Dessa forma, é esperado que o modelo vá se adequando ao ambiente através das interações durante o treinamento, fazendo com que o resultado produzido seja o mais próximo possível do valor exato.

Todo resultado obtido por um algoritmo com base em aprendizado supervisionado é avaliado para que seja possível julgar se a resposta está correta ou não. Assim, no fim, é comum não obter respostas concretas acerca de uma questão qualquer utilizando um algoritmo de Aprendizado de Máquina, o que se tem é uma porcentagem indicando o grau de confiança sobre uma resposta.

Uma vez que aprendizado supervisionado consiste em induzir os resultados obtidos por um algoritmo para um valor conhecido, o aprendizado não supervisionado obtém a melhor resposta através da experiência obtida durante o treinamento, sendo possível avaliar o comportamento em decorrência das conclusões obtidas anteriormente.

Um exemplo prático seria descobrir cada configuração de um tabuleiro de xadrez em que a posição das peças brancas é melhor que a posição das peças pretas Shalev-Shwartz e Ben-David (2014). Nesse exemplo, não é possível obter o conhecimento utilizando aprendizado supervisionado dado que não há uma resposta tabulada para qualquer possibilidade.

Algoritmos com base em aprendizado não supervisionado, aplicados no exemplo acima, são capazes de aprender com cada possibilidade e no final formular um resultado. Não há necessidade de dados prévios como entrada, uma vez que o aprendizado ocorre com base na experiência obtida durante o treinamento.

Considerando os dois tipos de aprendizados mencionados, o próximo a ser apresentado pode ser entendido como a interseção entre aprendizado supervisionado e o aprendizado não supervisionado. Este é definido como aprendizado semi-supervisionado e suas características são formadas em parte pelo aprendizado supervisionado e o aprendizado não supervisionado.

O principal objetivo do aprendizado semi-supervisionado é preencher as desvantagens encontradas tanto no aprendizado supervisionado quanto no aprendizado não supervisionado. O aprendizado supervisionado requer uma enorme quantidade de dados de treinamento para classificar os dados de teste, o que é um processo econômico e demorado. Por outro lado o aprendizado não supervisionado não necessita de nenhum dado rotulado. (REDDY; PULABAIGARI; B, 2018)

O aprendizado semi-supervisionado busca obter vantagem através do uso inteligente dos dados que possuem algum rótulo para obter informações sobre o problema e utilizá-las para guiar o processo de aprendizado a partir dos exemplos não rotulados. (BRUCE, 2002).

Por fim, há também o aprendizado por reforço, onde este, por sua vez, é o aprendizado onde a experiência é obtida a partir da interseção com o ambiente e é utilizada para aprender uma política de atuação de maneira autônoma. A cada experimentação é possível aprender novas características sem que tenha algum exemplo fornecido previamente. (PERICO, 2012)

O agente aprendiz no aprendizado por reforço interage com o ambiente em intervalos de tempos divididos entre percepção e ação. O agente observa o estado do ambiente e escolhe uma ação para realizar Após realizar a ação, o agente recebe um sinal, onde este, por sua vez pode representar uma penalização ou uma recompensa, indicando o quão desejável é o estado resultante. (PERICO, 2012)

Uma vez que os conceitos iniciais necessários para a elaboração de métodos para detecção de anomalias foram apresentados, serão expostos a seguir definições mais específicas presentes na computação que permitirão construir técnicas para detecção de anomalias através do uso de redes neurais artificiais.

2.3 Redes Neurais Artificiais

O primeiro conceito fundamental é o de redes neurais artificiais, onde esse, no que lhe diz respeito, é um dos métodos mais efetivos no aprendizado em certos tipos de problemas. Por exemplo, o algoritmo backpropagation descrito ainda nessa seção que mostrou-se capaz de resolver problemas que envolvem reconhecimento de caracteres escritos à mão, aprendizado de palavras faladas e reconhecimento de faces. (MITCHELL, 1997)

As redes neurais artificiais foram inspiradas no sistema de aprendizado biológico humano que consiste em redes complexas formadas por neurônios interconectados. Simplificando o conceito, é possível entender redes neurais artificiais como uma rede

interconectada de unidades simples, onde essas recebem um valor como entrada e produzem um outro valor na saída. (MITCHELL, 1997)

Por mais que as redes neurais artificiais tenham sido inspiradas em redes neurais biológicas, há muitas complexidades que não são consideradas no modelo e outras que são consideradas inconsistentes ao comparar fielmente. Um exemplo dessa diferença é o fato das unidades simples das redes neurais artificiais possuírem como saída um único valor constante, quando o que ocorre nas redes neurais biológicas é a saída de valores temporais complexos.

A forma mais simples de rede neural presente na literatura é o Perceptron. O Perceptron é um algoritmo iterativo capaz de produzir um resultado através da combinação linear entre os valores de entrada x_1, x_2, x_3, ..., e um vetor de valores w_1, w_2, w_3, ..., que serão otimizados durante a execução. (ROSENBLATT, 1958)

O vetor de valores a serem otimizados no Perceptron é definido como peso, que por sua vez é responsável por determinar a contribuição de uma entrada x_i na saída do Perceptron. O valor resultante da combinação linear entre os dois vetores é utilizado junto com um limiar para definir qual será a saída do Perceptron. Caso o resultado seja maior que o limiar então a saída será 1, caso contrário -1.

Sendo mais preciso, dado uma entrada x_1 até x_n, a saída $\theta(x_1,...,x_n)$ computada pelo Perceptron é

$$\theta(x_1, ..., x_n) \begin{cases} 1, & \text{caso } w_0 + w_1x_1 + w_2x_2 + ... + w_nx_n > 0 \\ -1, & \text{caso contrário} \end{cases}$$

onde cada w_i é um valor real que representa o peso.

De modo a simplificar a notação matemática para o Perceptron, é possível considerar uma entrada constante $x_0 = 1$ que possibilita a formulação

$$\sum_{i=0}^{n} w_i x_i > 0, \tag{2.1}$$

ou em sua forma vetorial $\vec{w} \times \vec{x} > 0$. (MITCHELL, 1997)

O algoritmo Perceptron se inicia com valores arbitrários para seus parâmetros e são atualizados sempre que forem classificados erroneamente através dos ciclos de treinamento dos dados. Após um número finito de iterações o algoritmo encontra os valores para os parâmetros que permite obter os melhores resultados. (ANTHONY; BARTLETT, 2009)

O algoritmo Perceptron pode ser usado para representar diversas funções booleanas. Por exemplo, ao assumir valores booleanos para 1 e -1, verdadeiro e falso, respectivamente, é possível representar todas as funções booleanas primitivas: AND, OR, NAND e NOR. Contudo, não é possível implementar com apenas um Perceptron a função XOR, pois este algoritmo possui limitação para resolver problemas que não são linearmente separáveis. (MITCHELL, 1997)

Por mais que tenha sido mencionado que o Perceptron emite como resultado 1 ou -1, é possível que outros valores sejam utilizados. A saída de um Perceptron ou rede neural artificial no geral depende de uma função de ativação, onde esta define um limite o qual será utilizado para determinar o valor da saída.

2.3.1 Função de ativação

Função de ativação é definida como uma função que avalia o valor que será emitido na saída de um neurônio e é utilizada para introduzir um componente de não linearidade na rede neural. Tal valor pode ser utilizado como entrada em um outro neurônio ou como saída final do modelo. (OLGAC; KARLIK, 2011)

É possível encontrar na literatura diversas funções de ativação, onde essas são utilizadas em diferentes situações. As escolhas mais populares estão entre função sigmoide (sigm) e tangente hiperbólica (TanH). Para problemas onde é necessário o uso de ativações ilimitadas ou parcialmente limitadas, a função de ativação ReLU é mais utilizada. (TIMMONS; RICE, 2020)

A função de ativação sigmoide é geralmente utilizada em problemas de regressão logística. Essa função de ativação é definida como

$$\sigma(x) = \frac{1}{1 + e^x} \tag{2.2}$$

o que permite com que o valor da saída esteja entre 0 e 1. Essa função de ativação permite com que a saída de um neurônio esteja entre 0 e 1. (You et al., 2020)

Apesar de ser encontrada em trabalhos na literatura, a função sigmoide possui uma limitação que a impede de ser amplamente utilizada sem grandes preocupações. Quando uma entrada é excessivamente grande ou pequena, o gradiente de um neurônio fica próximo de 0. Esse erro pode se tornar agravante após a propagação do valor para outro neurônio da rede. (You et al., 2020)

É possível definir a função de ativação tangente hiperbólica como sendo uma função obtida a partir de modificações realizadas na função sigmoide, ocasionando em uma mudança nos valores de ativação de um neurônio, -1 e 1, desativado e ativado, respectivamente. pesar das duas funções serem semelhantes, na prática, algumas aplicações possuem resultados superiores com a função de ativação tangente hiperbólica.

A função de ativação tangente hiperbólica possui a mesma limitação encontrada na função sigmoide. Quando a entrada é muito grande ou muito pequena, a derivada é bastante próxima de 0, fazendo com que a rede não sofra nenhuma alteração nos seus valores.

Por outro lado, há a função de ativação ReLU, definida como:

$$f(x) = max(x, 0) \tag{2.3}$$

Essa função de ativação permite com que as redes neurais sejam mais facilmente otimizadas do que as redes com a função de ativação sigmoide ou tangente hiperbólica. Este fato ocorre pois sua fórmula é bastante simples e a alteração da rede durante o treinamento ocorre de maneira mais fluida. Ambos os fatores fizeram com que a função $ReLU$ se tornasse padrão no aprendizado profundo. (RAMACHANDRAN; ZOPH; LE, 2017)

Apesar do grande uso da função *ReLU*, ainda é possível encontrar algumas dificuldades durante o treinamento de uma rede neural. Quando uma entrada da função for negativa, os pesos na rede neural não são atualizados, fazendo com que a o treinamento fique lento e alguns neurônios ineficazes. (You et al., 2020)

Levando em consideração o benefício do uso da função de ativação *ReLU* e seu problema, algumas melhorias foram desenvolvidas permitindo a criação de duas novas funções denominadas *Leaky ReLU* e Parametric ReLU. A função *Leaky ReLU* possui o uso de um valor 0.01 de correção que garante uma pequena linha reta na função de ativação. (You et al., 2020)

A função Parametric ReLU é uma generalização da função *Leaky ReLU*, permitindo parametrizar o valor de correção denominado neste caso por α. O parâmetro de correção pode ser aprendido junto com outros valores da rede neural durante o processo de treinamento. As fórmulas das funções Leaky ReLU e Parametric ReLU são representadas nas equações 2.4 e 2.5, respectivamente.

$$f(x) = max(0.01x, x) \tag{2.4}$$

$$f(x) = max(\alpha x, x) \tag{2.5}$$

2.3.2 Treinando de Redes Neurais Artificiais

O aprendizado em uma rede neural artificial está relacionado diretamente com o treinamento aplicado nela. O objetivo do treinamento em uma rede neural é permitir com que o vetor de pesos da rede possua os melhores valores para o contexto em que está sendo aplicado. Tais pesos são definidos através do uso de exemplos aplicados em cada iteração da fase de treino, possibilitando a análise da qualidade do valor obtido na saída.

Uma das formas de obter valores corretos para os pesos é com a inicialização do vetor com números aleatórios e realizar atualizações em cada iteração do treinamento

conforme necessário. A atualização dos pesos ocorre até que a rede seja capaz de produzir resultado aceitável em todos os exemplos utilizados nas iterações, fazendo com que o tempo necessário para o tempo de uma rede seja variável.

Na Figura 2.3 está sendo exibido um exemplo de neurônio junto com uma função de ativação, assunto que será tratado mais a frente, e o uso de um bias. O bias é um parâmetro utilizado na rede neural que permite com que os valores se adaptem ao conjunto de hipóteses do problema. (MITCHELL, 1997)

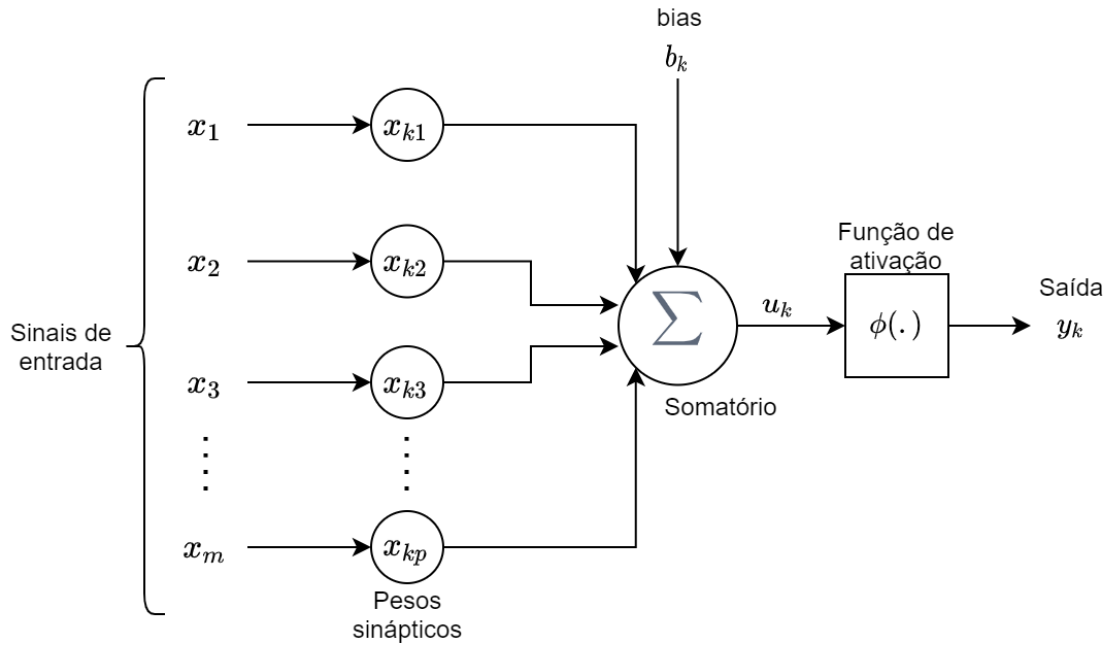


Figura 2.3: Modelo de um neurônio simples contendo m entradas e obtendo um valor na saída.

Quando se trata de redes neurais artificiais há algumas questões fundamentais acerca do resultado obtido. Durante o treinamento é preciso ter garantia de que os dados iniciais utilizados como hipótese contenham o valor objetivo da rede. Caso contrário não será possível garantir que a rede obterá respostas corretas e ,além disso, terá boa capacidade de generalização para o uso em instâncias que não foram observadas.

Essas e outras questões são analisadas no desenvolvimento de uma rede neural, principalmente durante o treinamento, uma vez que este é o momento onde a rede vai se desenvolver possibilitando a geração de resultados próximos ou iguais ao valor objetivo independente da situação, observadas ou não.

Há diversos algoritmos que podem ser utilizados para realizar o treinamento de uma rede neural artificial. Como dito em Mitchell (1997), os pesos no Perceptron são alterados em cada iteração seguindo sua regra de treinamento, onde esta, atualiza um peso w_i associado à entrada x_i como nas equações a seguir:

$$w_i \leftarrow w_i + \Delta w_i \tag{2.6}$$

onde

$$\Delta w_i = \eta(t - \theta)x_i \tag{2.7}$$

Considerando as equações 2.6 e 2.7, a variável t é o valor objetivo que deve ser alcançado baseando-se no exemplo atual da iteração, o θ é a saída gerada pelo Perceptron naquele momento e η é um número positivo denominado taxa de aprendizado. A taxa de aprendizado é utilizada para controlar o quanto um peso da rede precisa ser alterado em cada iteração do treinamento.

A convergência dos pesos durante o treinamento é um fato essencial. Suponha, considerando o algoritmo Perceptron, que a saída para um exemplo seja -1 e o valor verdadeiro seja 1. É necessário fazer com que os pesos da rede se alterem de modo que o valor de $\vec{w} \times \vec{x}$ aumente, permitindo que seja possível obter como resultado final o valor 1.

Como exemplo de convergência dos valores, temos: $x_i = 0.8$, $\eta = 0.1$, $t = 1$ e θ = -1, o peso atualizado será $\Delta w_i = \eta$ (t - θ)$x_i = 0.1(1 - (-1))0.8 = 0.16$. Esse resultado é utilizado na equação 2.6, permitindo que o peso aumente o valor e, dessa forma, possibilita com que o resultado final seja mais próximo do objetivo. Por outro lado, caso t = -1 e $\theta = 1$, os pesos associados a um número positivo x_i diminuirão. (MITCHELL, 1997)

Apesar do procedimento de treinamento mencionado acima convergir após um número de aplicações da regra de treinamento do Perceptron, é necessário que os exemplos de treinamento sejam linearmente separáveis e seja usado um número pequeno para η. Caso os dados não sejam linearmente separáveis a convergência não

é garantida. (MITCHELL, 1997)

Levando em consideração as limitações das redes neurais artificiais com estrutura simples, surgiu a necessidade de buscar outras estruturas que permitissem a aplicação em problemas considerados mais complexos. É a partir dessa necessidade que as redes neurais multicamadas passam a ser utilizadas.

2.4 Redes Neurais Multicamadas

Rede Neural Multicamada é um tipo de arquitetura bastante referenciada pela literatura. Sua popularidade é explicada ao analisar sua principal característica: flexibilidade para obter soluções com qualidade para várias classes de problemas sem que seja necessário alterar radicalmente o modelo aplicado. (FEDERAL et al., 2014)

Nas redes neurais multicamadas os neurônios estão organizados em duas ou mais camadas de processamento. Sempre há uma camada de entrada e uma de saída em qualquer rede neural desse tipo. A camada de entrada possui o papel importante de distribuir cada entrada da rede nos neurônios da camada seguinte, os dados nessa etapa não são alterados pois o objetivo é apenas iniciar o processo de aprendizado da rede.

Uma forma simples desse tipo de rede neural é exibida na Figura 2.4(a), nela há uma rede contendo 3 neurônios que recebem os valores de entrada entradas e 4 neurônios na saída. Percebe-se nesse exemplo que é possível manipular a quantidade de neurônios utilizados durante o desenvolvimento da rede.

Além da camada de entrada e camada de saída, as redes neurais com múltiplas camadas podem apresentar outras camadas em seu interior, estas são conhecidas como camadas ocultas. A utilização dessas camadas internas faz com que muitas limitações apresentadas pelo Perceptron deixam de existir. A Figura 2.4(b) apresenta uma rede neural multicamada com uma camada oculta.

Ao considerar redes multicamadas é preciso entender dois conceitos fundamentais que permitem a propagação de valores por todas as camadas da rede. Esses conceitos

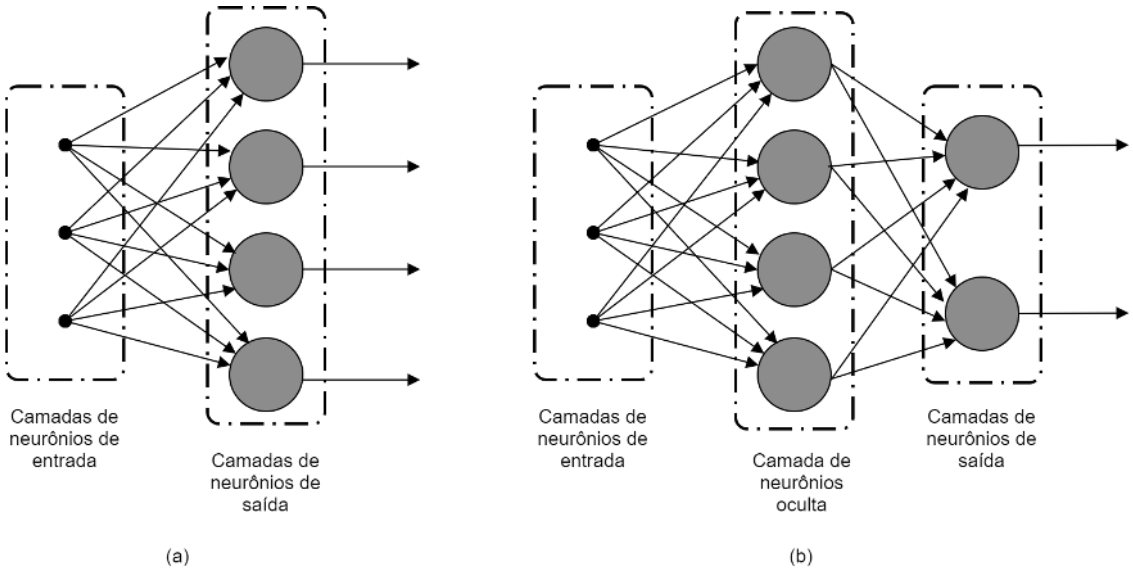


Figura 2.4: Redes neurais em camadas: (a) rede neural com apenas duas camadas de neurônios, (b) rede neural com uma camada de entrada, uma camada oculta e uma camada de saída

são: Forward Propagation e Backpropagation. Ambas ideias possibilitam com que qualquer rede neural que possua duas ou mais camadas consiga obter melhores valores para seus pesos durante o treinamento.

2.4.1 Forward Propagation

Esta etapa da rede neural consiste em propagar para a camada sucessora as informações que foram processadas pelos neurônios da camada atual. Esse processo ocorre desde a camada inicial até a produção de um resultado na camada de saída, fazendo com que a informação passe por toda a rede e consiga gerar aprendizado. (FEDERAL et al., 2014)

A informação que passa de uma camada para a outra possui dependência com o tipo de função de ativação utilizada, onde esta, por sua vez, pode variar de acordo com o propósito da rede neural, podendo ser qualquer uma das que foram apresentadas anteriormente, ou alguma outra presente na literatura. A escolha da função de ativação varia de acordo com a natureza do problema abordado.

A Figura 2.5 exemplifica uma rede neural com 3 camadas, sendo: uma camada inicial, uma camada oculta e uma camada de saída. Ambos os neurônios na camada

de entrada possuem o valor 1, em seguida, é possível observar os pesos que cada neurônio utilizará para o cálculo do valor que será usado como entrada no próximo neurônio, tal cálculo está explicitado na equação 2.8.

$$f(x) = \sigma(\sum_{i=1}^{n} x_i w_i) \tag{2.8}$$

Na equação 2.8, x_i representa a entrada i, w_i é o peso associado à entrada i e σ á uma função de ativação. Percebe-se que essa equação é bastante semelhante a equação 2.1, tendo como diferença somente o incremento da função de ativação diretamente no cálculo.

Na Figura 2.5 há uma rede com 3 camadas: camada inicial, camada interna e camada de saída. Nesse exemplo, como pode ser visto em Ayyadevara (2019), os pesos já foram definidos inicialmente de forma aleatória. Os valores dos neurônios da camada interna e camada de saída são calculados de acordo com a equação 2.8.

Seguindo com o mesmo exemplo da Figura 2.5, considere h1, h2 e h3 como o valor para cada neurônio da camada interna. O cálculo desses valores ocorrem da seguinte forma:

$$h_1 = 1 * 0.8 + 1 * 0.2 = 1$$

$$h_2 = 1 * 0.4 + 1 * 0.9 = 1.3$$

$$h_3 = 1 * 0.3 + 1 * 0.5 = 0.8$$

A Figura 2.6 exibe os valores dos neurônios da camada interna calculados. Após esse ponto, como visto anteriormente, há o uso de uma função de ativação para avaliar o resultado obtido e, dessa forma, promover uma saída. Neste exemplo será utilizado a função de ativação Sigmoide. Os valores com o uso da função de ativação ficam como segue:

$$final\_h_1 = S(1.0) = 0.73$$

$$final\_h_2 = S(1.3) = 0.78$$

$$final\_h_3 = S(0.8) = 0.69$$

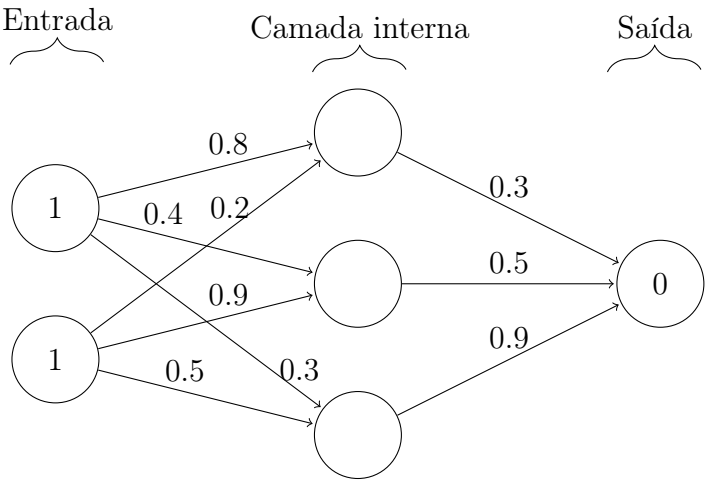


Figura 2.5: Propagação de informações da camada inicial até a camada de saída

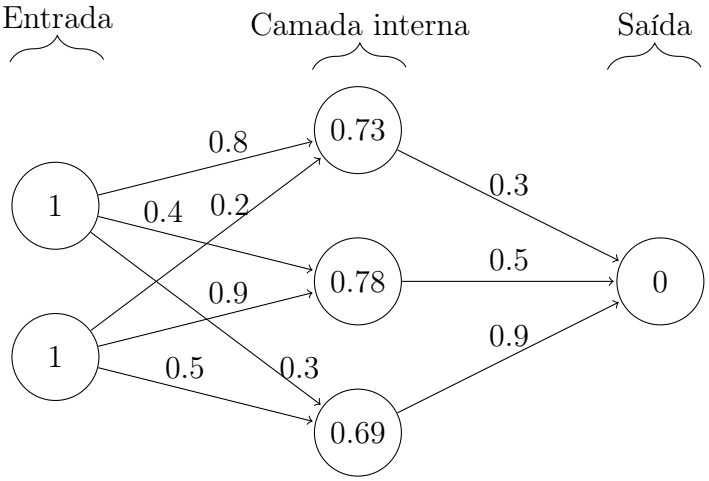


Figura 2.6: Neurônios da camada interna com valores calculados.

Os valores calculados para a camada interna serão utilizados como entrada na camada de saída, como visto na Figura 2.7. Da mesma forma que o cálculo realizado para obter os valores da camada interna, o valor final obtido pela rede é calculado da seguinte forma:

$$0.73 * 0.3 + 0.79 * 0.5 + 0.69 * 0.9 = 1.235$$

Como esperado, o valor obtido através dos cálculos na rede é diferente do valor desejado inicialmente, zero. Isso ocorreu devido a escolha aleatória dos pesos iniciais. Esse problema é resolvido através da atualização dos pesos obtidos no decorrer do treinamento da rede neural.

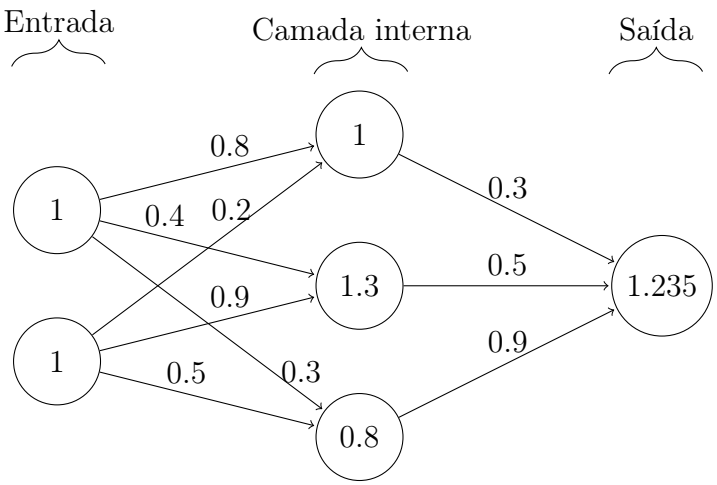


Figura 2.7: Neurônios da camada oculta com valores calculados.

Durante o treinamento da rede neural, um dos parâmetros mais importantes é o valor de perda. Tal valor tem o objetivo de indicar o quão longe o valor obtido na saída da rede está do valor objetivo. Geralmente quando a variável do problema é contínua, o valor da perda é calculado como o erro ao quadrado, porém há outras formas que podem ser exploradas. (AYYADEVARA, 2019)

Uma vez que a rede neural conseguiu produzir um resultado e o mesmo foi analisado, a atualização dos pesos, que outrora foram definidos de modo aleatório, é feita durante a fase denominada Backpropagation.

2.4.2 Backpropagation

Devido a existência do erro durante o aprendizado da rede neural, é necessário ter uma forma de contornar esse problema através de técnicas para redução do mesmo. Uma técnica que é utilizada é conhecida como Backpropagation. Esta por sua vez é a otimização dos pesos dos neurônios para que a rede neural consiga aprender o mapeamento das entradas para uma ou mais saídas (FEDERAL et al., 2014).

A atualização dos pesos é feita através do cálculo da derivada da função de perda considerando os pesos e o bias, esse cálculo é propagado recursivamente para as camadas anteriores, sendo uma camada por vez. Uma vez que uma camada na rede neural consiste em simples funções usando os valores da camada anterior, é possível utilizar a regra da cadeia para determinar as derivadas. O ajuste feito nos pesos dos

neurônios é determinado pelo gradiente da função de custo.

A derivada da função de custo indica a direção que a função está seguindo, lembrando que o objetivo é manter os valores dessa função o mais baixo possível. Já o gradiente da função de custo indica a variação que o peso de um neurônio precisa ter, considerando tanto mudança positiva quanto negativa, de modo que a função de custo seja mínima.

Em cada execução do algoritmo Backpropagation os pesos dos neurônios são alterados a partir da seguinte equação:

$$W = W - \alpha \frac{\partial J}{\partial W} \tag{2.9}$$

Observando a equação 2.9 é possível obter algumas informações acerca do comportamento da rede neural em relação a busca pelos melhores valores para os pesos de cada neurônio.

Caso a derivada da função de custo seja positiva, significa que um aumento no peso resultará em um aumento no erro, portanto, o novo peso a ser atualizado precisa ser menor. Se a derivada da função de custo for negativa, será necessário aumentar aumentar os pesos dos neurônios, uma vez que dessa forma será possível reduzir o erro.

Além dos casos onde é necessário atualizar os pesos dos neurônios para obter o menor erro possível, há também a situação onde o valor da derivada da função de custa é igual a zero, nesse caso, o estado atual é referente ao mínimo estável, logo, não é necessário alterar o valor do peso.

A taxa de aprendizado α é introduzida na equação 2.9 como uma constante, a fim de que o peso a ser atualizado seja de forma suave. O uso do α evita problemas de grandes saltos entre os valores, fazendo com que a alteração dos pesos dos neurônios não ultrapasse o valor capaz de gerar a solução com menor custo. A medida utilizada para α é, comumente, bem pequena, mas no geral ela é ajustável a cada tipo de rede. (JACOBS, 1988)

Aproveitando as características e vantagens das redes neurais multicamadas há também as redes neurais convolucionais. Estas por sua vez possuem características e particularidades que as tornam bastante úteis quando o dado processado é uma imagem.

2.5 Redes Neurais Convolucionais

Rede Neural Convolucional, ConvNet ou CNN, é um tipo de rede que devido as suas propriedades é utilizada para processamento de dados na forma de imagens e no reconhecimento de objetos. Uma Rede Neural Convolucional consegue identificar os objetos de uma imagem de entrada e atribuir uma configuração para os pesos da rede de modo que cada aspecto da imagem consiga ser diferenciável.

Diferente das redes neurais convencionais, as redes neurais convolucionais não precisam de um pré-processamento longo, pois possuem capacidade de aprender, de modo independente, as características dos dados. Apesar dessa vantagem, uma CNN exige maior cuidado na escolha dos filtros que serão utilizados.

Em uma rede neural convolucional há o uso de filtros, onde esses são estruturas usadas para identificar os padrões das imagens, sendo possível realizar diversas configurações nos parâmetros possibilitando obter bons resultados em diferentes situações. Em cada filtro, cada neurônio está conectado a apenas um subconjunto dos neurônios da camada anterior. (FEDERAL et al., 2014)

Considerando o processamento de imagens, é possível realizar certo tipo de manipulação na imagem para utilizar um Perceptron multicamada. Uma imagem é entendida como uma matriz, mas é possível fazer uma transformação para vetor, como visto na Figura 2.8, e utilizá-lo como entrada para o Perceptron.

O problema do uso de uma imagem em sua forma vetorial é que não pode ser aplicado em todos os casos. Ao analisar imagens binárias e simples, o método consegue ter um desempenho médio na identificação das classes, porém, em casos com imagens mais complexas o desempenho é quase nulo devido as limitações do modelo Perceptron.

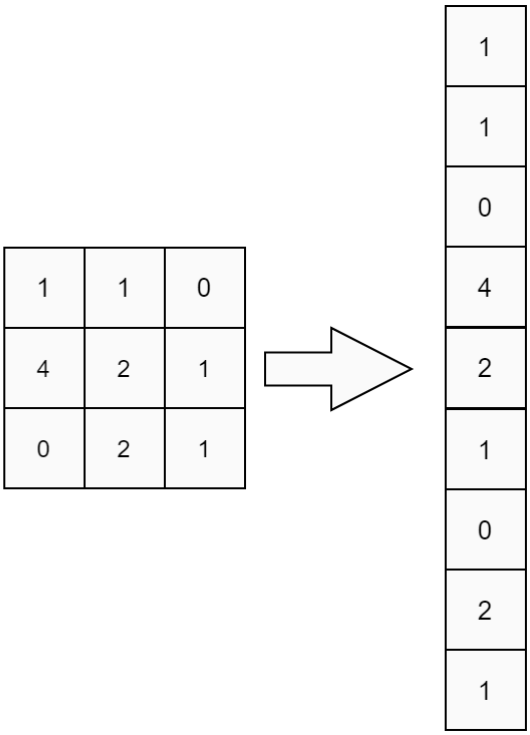


Figura 2.8: Transformação de uma matriz 3x3 em um vetor 9x1

Uma rede neural convolucional é uma ferramenta com capacidade de capturar as dependências espaciais e temporais em uma imagem através do uso dos filtros. Com essa arquitetura de camadas convolucionais é possível fazer com que a rede consiga obter melhores resultados durante o treinamento. Para isso, é necessário configurar cada camada da rede de modo que o ganho seja maximizado.

2.5.1 Camadas Convolucionais

Em (LECUN et al., 1989), camadas convolucionais possuem filtros treináveis que podem ser aplicados por toda entrada. Nas entradas no formado 2D, ou seja, imagens, os filtros são definidos como uma pequena matriz de pixels e os neurônios são conectados apenas aos outros neurônios que estiverem compreendidos na área do filtro da camada anterior.

Durante o treinamento da rede, os filtros são executados na entrada e geram uma valor, onde este, por sua vez, compõe a saída. A equação que define uma saída de um filtro está disposta na equação 2.10.

$$y_{rc}^{l} = \sum_{i=1}^{F_r} \sum_{j=1}^{F_c} \sigma(y_{(r+i-1)(c+j-1)}^{l-1} w_{ij}^{l} + b^{l}) \tag{2.10}$$

Na equação acima, y_{rc}^{l} é a saída da posição $\{r, c\}$, F_r e F_c representam o número de linhas e colunas no filtro, respectivamente, w_{ij}^{l} é o valor do filtro na posição $\{i, j\}$, $y_{(r+i-1)(c+j+1)}^{l-1}$ representa o valor da entrada para essa camada na posição $\{r + i - 1, c + j - 1\}$ e b^{l} é o bias.

A equação 2.10 é definida para todas as possíveis aplicações de filtro que estejam no intervalo permitido, ou seja, $r \in \{1, 2, ..., X_r - F_r + 1\}$ e $c \in \{1, 2, ..., X_c - F_c + 1\}$, onde X_r e X_c representam o número de linhas e colunas da entrada nessa camada, respectivamente, e F_r e F_c representam o número de linhas e colunas no filtro, respectivamente.

Os filtros podem ser aplicados em todos os possíveis pares de entradas $\{r, c\}$ da camada convolucional, porém, por questões de tempo computacional, uma estratégia utilizada é considerar apenas pares com uma certa distância s, esta por sua vez é conhecida como *stride*. Ter uma distância igual a 1, ou seja, $s = 1$ é o mesmo que aplicar o filtro por todos os possíveis pares.

A Figura 2.9 representa a forma matricial de uma imagem qualquer que será utilizada como entrada nas convoluções da rede. A imagem está representada como a junção da matriz em verde com a matriz em laranja.

Em seguida, a matriz em laranja também na Figura 2.9, representa o filtro que será utilizado para os cálculos oriundos da equação 2.10. Por fim, a Figura 2.10 exibe a matriz de saída preenchida dessa etapa. Lembrando que, o valor nulo foi considerado para o bias, ou seja, zero. Para esse exemplo foi utilizado stride = 1, ou seja, todos os pares foram considerados.

Durante o processo, o filtro se move da esquerda para a direita da imagem considerando o valor do stride, até alcançar o valor relativo a largura da imagem. Nesse momento o filtro volta para o início da imagem definido pelo lado esquerdo e, considerando o mesmo stride, move-se para baixo para repetir o mesmo processo

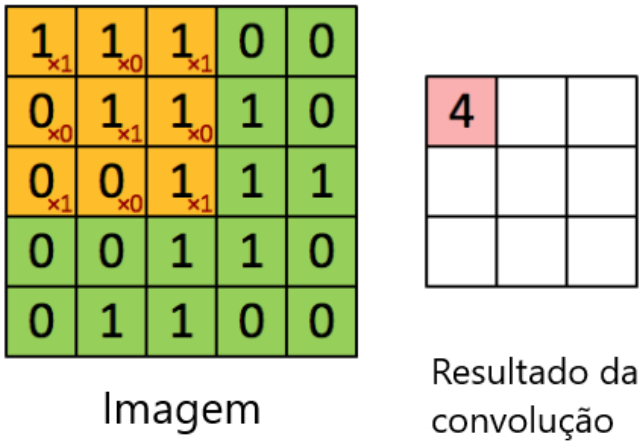


Figura 2.9: Exemplo de convolução utilizando um filtro 3x3

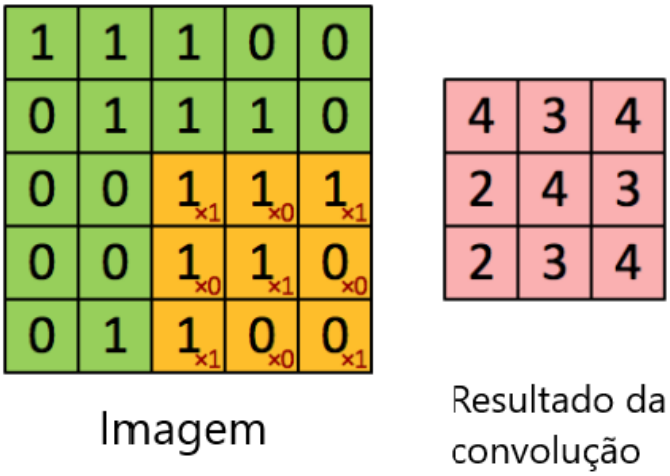


Figura 2.10: Resultado da convolução completa a partir de um filtro 3x3

anterior. Esse processo ocorre até que o filtro percorra toda imagem.

Além das camadas convolucionais mais simples com o uso de filtros para identificar partes importantes de uma imagem, redes neurais convolucionais também contêm camadas de agrupamento (*Pooling Layers*). Essas camadas são geralmente usadas após as camadas convolucionais e têm como objetivo simplificar as informações na saída da camada convolucional.

2.5.2 Pooling Layers

Durante as convoluções de uma rede neural convolucional as informações são simplificadas de uma camada para outra. Esse processo ocorre através do uso da camada de agrupamento, permitindo que somente a unidade que for relevante seja passada para a próxima camada convolucional.

Por exemplo, caso a saída de uma camada convolucional seja 32x32, então a saída de uma camada de agrupamento inserida logo após a camada que gerou a imagem 32x32 será 16x16, caso o fator de redução seja 2. Na literatura, é possível encontrar diferentes estratégias para manipulação dos dados, mas o mais utilizado é o max-pooling. (FEDERAL et al., 2014)

A equação 2.11 representa a formulação matemática para max-pooling. Nessa equação, y^l_{rc} reflete a saída para o índice $\{r, c\}$, m é o tamanho da área de agrupamento e $y^{l-1}_{(r+i-1)(c+j-1)}$ é o valor da entrada na posição $\{r + i - 1, c + j - 1\}$. Da mesma forma que é possível utilizar um stride nas camadas convolucionais, o mesmo ocorre nas camadas de agrupamento.

A Figura 2.11 exibe a etapa de propagação do max-pooling com stride=1. Como visto, somente o maior valor do filtro é passado adiante, dessa forma, entende-se que apenas as características mais marcantes são consideradas, fazendo com que detalhes mais simples sejam deixados de lado uma vez que não irão ter grande impacto no resultado final. (FEDERAL et al., 2014)

$$y^l_{rc} = \max_{i,j \in \{0,1,\ldots,m\}} y^{l-1}_{(r+i-1)(c+j-1)} \tag{2.11}$$

Redes neurais convolucionais são ótimas ferramentas para tratamento de imagens. As etapas que ocorrem através das camadas convolucionais são capazes de aprender características relevantes da imagem e produzir um resultado na saída da rede. Essas características fazem com que esse tipo de rede neural seja eficiente para metodologias envolvendo detecção de anomalias.

Seguindo com a ideia de detecção de anomalias, é necessário o uso de alguma

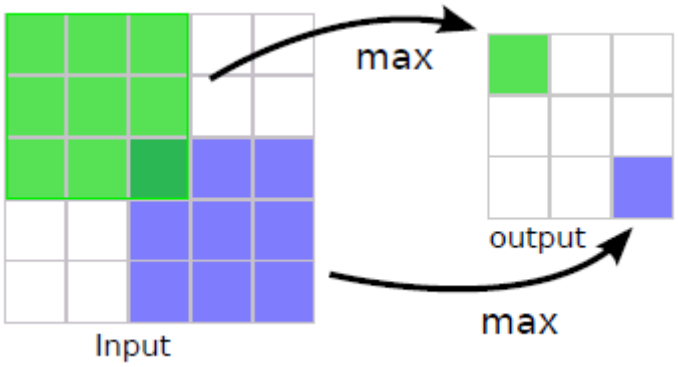


Figura 2.11: Fase de propagação de uma camada de agrupamento utilizando o max-pooling

técnica que seja capaz de ter uma imagem como dado de processamento e produzir uma saída com base no que foi aprendido. Para isso, uma das técnicas populares acerca de tratamento de imagem é o Autoencoder onde este, por sua vez, será melhor abordado na seção seguinte.

2.6 Autoencoder

Autoencoder é um tipo de rede neural artificial de aprendizado não supervisionado capaz de obter uma codificação a partir da redução da dimensionalidade de um conjunto de dados e depois reconstruí-los. Seu principal objetivo é aprender uma representação para um conjunto de dados através do treinamento da rede. (GONZALEZ, 2014).

O Autoencoder tenta aprender uma função $h_{W,b}(x) \approx x$, que por sua vez, representa aproximação para a função identidade dos dados envolvidos, fazendo com que uma saída $\bar{x}$ seja similar à entrada x. Essa técnica possui diversas aplicações, como por exemplo: redução de dimensão dos dados, remoção de ruídos em imagens, extração de características, geração de imagens, entre outras.

Variadas aplicações são possíveis, pois um Autoencoder pode assumir diferentes configurações dependendo do contexto da aplicação. Essa característica contribui para o surgimento de vários outros modelos derivados do Autoencoder convencional

como serão exibidos nas seções futuras.

Um Autoencoder é formado basicamente por três conceitos principais: Codificador, dados comprimidos e Decodificador. Cada parte que compõe o algoritmo Autoencoder possui funções únicas que permitem com que as principais características dos dados de uma imagem sejam aprendidas, de modo que na saída da rede neural seja possível obter um resultado de boa qualidade.

O codificador é a etapa responsável por aprender todas as características que configuram o dado e gerar uma representação codificada dos mesmos. Durante o processo de codificação, as dimensões são reduzidas, provocando leves perdas de dados, podendo também causar perdas de algumas informações na imagem final.

A camada dos dados comprimidos contém os dados com a menor dimensão definida pelo modelo. Nessa parte, os dados já estão codificados com todas as características importantes obtidas durante a fase de codificação. Dessa forma, o tamanho dos dados atual se apresenta bem menor quando comparado ao tamanho original.

O decodificador é responsável por aprender a reconstruir os dados a partir da codificação gerada na primeira etapa. É compreensível caso o dado reconstruído não seja idêntico ao original pois como mencionado anteriormente, durante o processo de codificação os dados têm suas dimensões reduzidas e isso faz com que algumas informações possam ser perdidas. Porém, características importantes das imagens são mantidas o que possibilita manter semelhanças com a imagem original.

A rede responsável por codificar o dado pode ser representada pela função de rede neural convencional. Tal função faz o uso de valores de entrada, pesos da rede e o bias, além da função de ativação que de fato decidirá como a informação será passada adiante na rede. Essa função é exibida na equação 2.12. Já a decodificação pode ser representada da mesma forma, porém com diferentes pesos, bias e função de ativação como visto na equação 2.13.

$$z = \sigma(Wx + b) \tag{2.12}$$

$$x' = \sigma'(W'z + b') \tag{2.13}$$

Assim como em outras redes neurais, o aprendizado em um Autoencoder ocorre durante a fase de treinamento. O diferencial nesse caso é que, geralmente, o erro utilizado como critério para o fim do treinamento é diferença entre a imagem de entrada e a imagem de saída. Logo, o algoritmo tenta ao máximo obter uma representação fiel da imagem original.

Como visto na Figura 2.12, o dado em questão se trata de uma imagem, onde é possível observar bastante semelhança entre a imagem de saída e a imagem de entrada. Como o algoritmo baseia-se em compressão de dados, é esperado que tenha pelo menos uma sutil diferença ao comparar com o dado original, porém mesmo com essa limitação o Autoencoder consegue gerar bons resultados em suas aplicações.

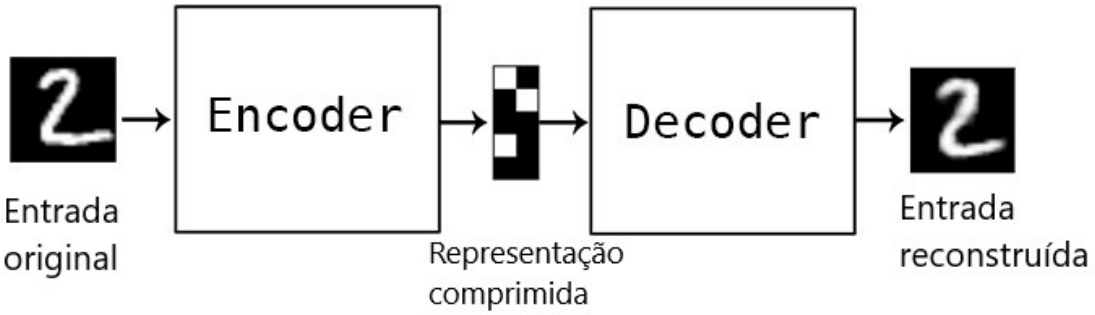


Figura 2.12: Exemplo da estrutura de um Autoencoder utilizando uma imagem simples

Assim, como as redes neurais não convolucionais possuem limitações ao analisar imagens complexas, o mesmo ocorre com o Autoencoder convencional. Tendo em vista o problema mencionado, torna-se necessário buscar uma variação do algoritmo Autoencoder que consiga obter resultados ainda melhores ao considerar imagens com características complexas. Desta forma, é apresentado então o modelo Autoencoder convolucional.

2.6.1 Autoencoder Convolucional

Autoencoder Convolucional é uma variação do Autoencoder convencional capaz de gerar resultados satisfatórios quando utilizado com imagens mais realistas. Isso ocorre devido ao uso das propriedades das redes neurais convolucionais e a capacidade de compartilhar os pesos por todas as entradas, preservando a localidade espacial dos dados (TURCHENKO; LUCZAK, 2015).

Assim como o Autoencoder, o Autoencoder Convolucional é baseado no paradigma codificador-decodificador. A entrada é comprimida na fase de codificação e expandida na fase de decodificação, de modo que seja possível obter a imagem inicial ou uma aproximação da mesma.

A arquitetura de um Autoencoder convolucional é semelhante a que foi mencionada na seção 2.5, diferenciando apenas com o uso do compartilhamento dos pesos. Considerando uma entrada x com apenas um canal, a representação latente para a k-ésima característica latente é dada por

$$h^k = \sigma(W^k x + b^k) \tag{2.14}$$

onde o bias é compartilhado por todo o mapeamento, σ é a função de ativação, onde está pode ser alguma descrita na seção 2.3.1 ou alguma outra.

As camadas de um Autoencoder Convolucional podem ser entendidas como um mapa com as características da imagem utilizada na entrada. Cada unidade desse mapa está conectada a um ponto na camada anterior, permitindo com que características distintas sejam consideradas durante o desenvolvimento do modelo. Somente um bias é utilizado nas camadas pois o uso de um bias por pixel permitiria muitas variações, tornando o modelo bem mais complexo. (MASCI et al., 2011)

A reconstrução da imagem após a fase de codificação é feita a partir da seguinte equação:

$$y = \sigma(\sum_{k \in H} W^k h^k + c) \tag{2.15}$$

Considerando a equação 2.15, c representa o bias, h identifica o grupo de características latentes e W representa a operação de inversão nas duas dimensões dos pesos. (MASCI et al., 2011)

Geralmente no uso do Autoencoder convolucional, a função de custo mais comum é o erro quadrático médio (MSE), definido como:

$$E(\theta) = \frac{1}{2n} \sum_{i=1}^{n} (x_i - y_i)^2 \tag{2.16}$$

Levando em consideração o resultado que a técnica do Autoencoder Convolucional é capaz de fornecer, torna-se viável seu uso na elaboração de ideias que possibilitem a detecção de anomalias, uma vez que os detalhes presentes na imagem são possuem maior relevância. Com o Autoencoder Convolucional é possível obter um modelo capaz de aprender as características das imagens de forma que o conceito de anomalia seja compreendido.

Além do Autoencoder Convolucional é necessário ter um mecanismo capaz de julgar a possível ocorrência de anomalia em uma imagem ou vídeo. Para isso, é utilizado um outro tipo de modelo que possui o objetivo de, a partir de um conjunto de dados, classificar os dados em determinados grupos. Esse tipo de modelo é denominado classificador, sua definição e características são apresentadas a seguir.

2.7 Classificador

Todos os dias o ser humano precisa tomar decisões, sejam elas simples ou complexas, onde estas podem ter grande impacto no resultado final. Essas decisões podem ser feitas através do auxílio de uma ferramenta de classificação para maior precisão com base em algum parâmetro. Levando em consideração o ramo profissional, as empresas possuem diversos problemas de classificação, ocorrendo em áreas de pesquisa até tomadas de decisão que decidem o futuro no mercado. (DEY et al., 2014)

Conceituando, um classificador é qualquer algoritmo que consegue agrupar dados em classes distintas. Um exemplo prático são os filtros feitos em e-mails que

categorizam cada e-mail novo como spam ou não spam. Essa classificação pode seguir diferentes estratégias, dependendo de como o algoritmo foi construído, porém no geral é uma implementação de reconhecimento de padrões.

Para classificação não há uma especificidade de qual tipo de dado deve ser utilizado. O classificador consegue manipular qualquer dado desde que o conjunto apresente características semelhantes, ou não, sendo nesse caso uma classe para cada dado diferente. Uma vez que este trabalho trata-se de técnica para classificação de anomalias em vídeo, os dados que serão considerados são imagens.

Sabendo que um vídeo é conceituado como um sequência de imagens, é necessária a existência de uma forma capaz de analisar cada imagem de maneira rápida e eficiente. Nesse caso a eficiência está relacionada as dificuldades que o dado pode apresentar, tais como: presença de ruídos, qualidade baixa, oclusão de objetos e outros. (DEY et al., 2014)

O principal objetivo da classificação de imagens é identificação precisa dos objetos que estão em cena. A classificação de imagem faz o uso tanto de classificação supervisionada quanto não supervisionada. De maneira análoga ao aprendizado supervisionado e não supervisionado, na classificação supervisionada os dados utilizados precisam estar rotulados e esta tarefa é feita por um humano. Já na classificação não supervisionada, toda a operação de classificar as imagens é feita pelo computador.

Assim como qualquer outro modelo de rede neural, um classificador precisa passar pela etapa do treinamento, onde este é abordado de duas formas: supervisionado e não supervisionado. Na classificação somente supervisionada é necessário ter conhecimento prévio das imagens quer serão consideradas pelo classificador antes de executar o treinamento. A vantagem dessa forma é que ao encontrar algum erro de classificação é possível corrigi-lo. A desvantagem está relacionada ao tempo gasto na rotulação dos dados e perda de características importantes por falha humana.

Já na classificação não supervisionada não é necessário ter conhecimento prévio dos dados e eles não precisam passar por intervenção humana antes do treinamento. Neste caso, os dados são agrupados através de características semelhantes com o

uso de algoritmos específicos. Sua principal vantagem está em ser rápido e livre de falhas humanas, porém dessa vez surgem os desafios provenientes da técnica de agrupamento de dados com base em características semelhantes.

Uma vez que foram apresentados o conceito e as características de um classificador, bem como os desafios presentes durante o treinamento, é possível iniciar a abordagem da técnica de classificação de anomalias presente neste trabalho.

Capítulo 3

Classificação automática de anomalias em vídeo

Neste capítulo será descrita a solução proposta por este trabalho para a detecção automática de anomalias em vídeo a partir do uso de redes neurais artificiais. Inicialmente será exposta a motivação pela qual a técnica foi elaborada, em seguida, alguns trabalhos presentes na literatura com aplicação nesse contexto e ,por fim, a exposição dos modelos utilizados bem como a lógica que os relacionam.

3.1 Introdução

A classificação automática de anomalias em vídeo é um problema explorado na literatura devido as diversas possibilidades de aplicações em diferentes contextos. Serviços de segurança e monitoramento são alguns dos principais exemplos visto o grande avanço que pode haver no desempenho de detecção de comportamentos anômalos.

Entende-se como anomalia qualquer comportamento que não esteja de acordo com o estabelecido para um determinado ambiente naquele momento, uma vez que o padrão comportamental pode variar com o decorrer dos anos. Dessa forma, anomalias em vídeos são ocorrências detectadas a partir de vídeos de diferentes situações, como

por exemplo: câmera de segurança de parques, rodovias, bancos, entre outros.

A análise manual de um vídeo com o intuito de identificar anomalias é um processo que, dependendo do contexto, pode ser bastante demorado e cansativo. Como dito em Varghese, Mulerikkal e Mathew (2017), há diferentes casos de anomalias, onde estes podem variar em grau de urgência, como por exemplo: acesso não autorizado a um local específico, eventos anormais que caracterizam tentativas de roubo ou ataque terrorista, acidentes de trânsito.

Em casos onde há uma certa urgência na identificação da anomalia, como o caso em câmeras de segurança, a detecção de forma manual não é uma opção interessante. Nessa ocorrência, é necessário ter uma forma mais eficaz e rápida de detecção, uma vez que a análise humana ainda está sujeita a erros ocasionados por diferentes interpretações de um mesmo contexto. (VARGHESE; MULERIKKAL; MATHEW, 2017)

A tarefa de detectar anomalias em um vídeo que possui cenas complexas é considerado difícil em virtude das diferentes situações que podem ocorrer no mesmo momento. Nos vídeos é possível identificar grandes variações de comportamento, permitindo com que em um determinado segundo algo seja considerado anômalo e, no segundo seguinte, seja considerado normal.

Além das variações que ocorrem em uma cena, outro fator que possui influência na detecção de anomalia é o contexto histórico da região que está sendo retratada pelo vídeo. Com o decorrer dos anos o conjunto de regras de uma determinada região pode ser alterado, permitindo com que uma súbita mudança que não deve ser considerada como anomalia. Nesse caso ocorreu apenas uma mudança no comportamento considerado normal daquela região.

As mudanças que ocorrem em um determinado local não devem ser consideradas tão incomuns, uma vez que o grande dinamismo do mundo torna possível esse acontecimento. É impossível ter previamente todos os prováveis eventos de anomalias e mudanças que podem ocorrer, logo, é necessário ter uma análise elaborada acerca do contexto ao qual a técnica será aplicada.

A fim de evitar problemas com mudanças de comportamento na região, é necessário utilizar uma grande quantidade de dados para o treinamento da rede neural presente na técnica de classificação automática de anomalias. Dessa forma será possível abranger o maior número de possibilidades de anomalias anomalias, além de tornar o modelo mais confiável considerando os resultados.

Antes de apresentar a abordagem para detecção automática de anomalias presente neste trabalho, serão expostos alguns trabalhos relacionados que ajudarão a compreender melhor o estado atual da literatura acerca do problema em questão.

3.2 Trabalhos Relacionados

Considerando o benefício que a detecção de anomalias em vídeos possui para setores públicos e privados, muitos estudos foram realizados utilizando diferentes abordagens na tentativa de propor uma técnica eficaz para identificar anomalias em vídeos. Algumas dessas técnicas serão citadas a seguir.

Em Roshtkhari e Levine (2013), é apresentada uma técnica para detectar anomalias em tempo real. De acordo com esse trabalho, os quadros são separados em grupos e durante o processamento são utilizados conjuntos formados por alguns grupos de quadros relacionados. Em cada conjunto é calculada a probabilidade dos grupos de quadros serem considerados normais.

Caso um conjunto de um vídeo tenha bastante variação nas probabilidades dos grupos de quadros, este é separado como conjunto anômalo. O processo segue dessa forma até que todos os conjuntos do vídeo sejam analisados. Esta técnica baseia-se no aprendizado não supervisionado.

Outra técnica que também utiliza o aprendizado não supervisionado para identificação de eventos anômalos é descrita em Xu et al. (2015a). Neste trabalho, foi proposto um modelo capaz de aprender características e padrões dos movimentos dos objetos presentes em uma imagem, de forma que também seja possível identificar anomalias considerando o movimento realizado por um determinado objeto em um vídeo.

A detecção de eventos anômalos seguindo a técnica é feita com o uso de três algoritmos Support Vector Machine (SVM) de uma classe. Cada algoritmo SVM utilizará uma representação de características aprendida. Após o treinamento dos três algoritmos, cada um será capaz de produzir um valor que será combinado e utilizado na detecção de anomalia conforme os critérios utilizados no trabalho.

Em Sultani, Chen e Shah (2018) é proposto uma abordagem que, inicialmente, os vídeos são divididos em segmentos durante o treinamento, onde esses são classificados como anômalos ou normais. A estratégia principal consiste em tratar a detecção de anomalias como um problema de regressão, tendo como objetivo fazer com que segmentos com anomalias possuam tenham um valor maior quando comparado aos segmentos normais.

Além dos métodos de aplicações gerais, alguns trabalhos elaborados utilizam informações específicas do seu contexto para ajudar na detecção de anomalia, ou seja, eles possuem alguma informação sobre o evento anômalo que possibilite a técnica fique mais sofisticada, porém com o espaço de aplicação limitado.

Para esses casos mais específicos há os trabalhos de Datta, Shah e Lobo (2002), Gao et al. (2016), Kooij et al. (2015) e Mohammadi et al. (2016). Nos trabalhos mencionados, o objetivo é identificar ações violentas ou agressões humanas presentes em vídeos através da orientação dos movimentos e membros das pessoas.

A técnica utilizada em Kooij et al. (2015) consiste em usar frames e áudios como dados de processamento para detectar ações agressivas em vídeos de vigilância. Gao et al. (2016) propuseram métodos de extração de fluxo na detecção de violência em multidões. O mais recente mencionado proposto em Mohammadi et al. (2016) consiste em uma nova abordagem baseada em heurística de comportamento para classificar vídeos violentos e não violentos.

Além desses trabalhos, é possível encontrar também obras que utilizam o modelo Autoencoder para aprender o comportamento considerado normal e utilizar a perda obtida na reconstrução da imagem como método para detectar anomalia, como é o caso de Hasan et al. (2016) e Xu et al. (2015b)

Com base nestes trabalhos, será proposto um método composto por duas partes. Na primeira, será elaborado um modelo Autoencoder Convolucional com o objetivo de aprender características das imagens de forma que seja possível reproduzi-las considerando uma taxa de erro mínima. Na segunda etapa, será desenvolvido um algoritmo classificador responsável pela tomada de decisão indicando se há ou não a presença de uma anomalia. Mais informações a respeito das duas etapas mencionadas serão abordadas a seguir.

3.3 Método para classificação automática de anomalias em vídeos

O método para classificação automática de anomalias em vídeo é definido como uma técnica capaz de identificar a ocorrência de anomalia de maneira rápida e com alta porcentagem de certeza. Como foi visto nas seções anteriores, esta tarefa em contextos complexos é algo bastante complicado e podem ocorrer erros, logo, é preciso que o modelo seja confiante o suficiente para que a taxa de erro seja mínima.

A técnica apresentada nesse trabalho consiste em duas partes principais. Inicialmente, é usado um modelo Autoencoder Convolucional para aprender as características das imagens presentes no vídeo alvo e, em seguida, um modelo classificador, onde este será responsável por, de fato, categorizar como anomalia ou não.

Com o método para classificação automática de anomalias em vídeo, cada anomalia é detectada através do erro obtido na comparação entre a imagem obtida na saída do modelo Autoencoder Convolucional e a imagem original presente no vídeo analisado. Esse erro é utilizado no classificador como critério de decisão.

Durante o treinamento, o Autoencoder Convolucional aprenderá características dos objetos comuns no ambiente do vídeo. Ao utilizá-lo em vídeos que possuam imagens com um ou alguns objetos anômalos, é esperado que o Autoencoder não saiba reproduzi-lo da maneira correta. Dessa forma, a imagem resultante terá grande diferença em relação à quantidade de pixels presente na imagem original.

A seguir, serão apresentadas mais informações sobre como o Autoencoder Convolucional e o modelo classificador foram utilizados na detecção de anomalias em vídeo, bem como exemplos do desenvolvimento da ideia a partir do uso de imagens demonstrativas.

3.3.1 Autoencoder Convolucional

Como dito anteriormente, o Autoencoder Convolucional tem o papel de reproduzir imagens através do aprendizado de características principais dos objetos em uma cena. Dessa forma, é necessário que haja uma certa quantidade de vídeos que serão utilizados durante o treinamento da rede para que seja possível aprender a maior quantidade possível de objetos que estarão presentes em uma situação considerada normal, ou seja, sem ocorrência de anomalias.

Considerando o exemplo da Figura 3.1, é possível observar um caminho que deve ser utilizado somente por pedestres, logo, qualquer outro meio de locomoção como carro, bicicleta, skate e outros são estritamente proibidos. Nesses casos, os meios de locomoção proibidos são considerados anomalias para esse contexto e, precisam ser identificados.

Figura 3.1: Exemplo de uma imagem presente um vídeo onde há somente objetos normais ao contexto.

A Figura 3.2 representa uma anomalia encontrada no contexto do vídeo em questão. Nesse caso, há um veículo atravessando o caminho, caracterizando a ocorrência de uma anomalia. Nessa situação, durante o treinamento do Autoencoder Convolucional, as características não serão aprendidas, uma vez que essa imagem

não estará presente no conjunto de imagens de treinamento.

Ao utilizar a imagem da Figura 3.2 como entrada no Autoencoder Convolucional já treinado, todos os outros objetos que não forem anomalia serão representados com um pequeno erro causado durante a fase de codificação, exceto o veículo. Nesse caso, o Autoencoder Convolucional não saberá como representar esse objeto fielmente, fazendo com que o erro seja consideravelmente alto.

Figura 3.2: Exemplo de anomalia no mesmo contexto da Figura 3.1.

A Figura 3.3b representa a tentativa do Autoencoder Convolucional de reproduzir a imagem da Figura 3.3a que contém uma anomalia. É possível perceber que o modelo não soube representar o carro da maneira correta, há alguns detalhes que estão diferentes da imagem original. Os outros objetos foram exibidos sem grandes problemas mantendo o erro baixo.

(a) Imagem original presente no vídeo contendo uma anomalia.

(b) Reconstrução da Figura 3.3a utilizando Autoencoder Convolucional.

Figura 3.3: Autoencoder Convolucional reproduzindo uma imagem contendo uma anomalia

Para o desenvolvimento da técnica cada frame do vídeo é compreendido em duas

partes: a imagem original e a imagem gerada pelo Autoencoder Convolucional. É sabido que a imagem gerada pelo modelo Autoencoder possui algumas diferenças quando comparada com a imagem original, mas é necessário que essa diferença seja apresentada de uma forma que possa ser medida. Logo, o próximo passo é quantificar o erro obtido entre a imagem original e a imagem gerada pelo modelo.

O erro é calculado através da diferença absoluta entre a imagem obtida pelo modelo Autoencoder Convolucional e a imagem original presente no vídeo. Considerando a diferença entre a Figura 3.3a e a Figura 3.3b, é possível observar o erro através da Figura 3.4, nela há somente pixels que não foram reconstruídos pelo Autoencoder Convolucional.

Figura 3.4: Erro obtido a partir da diferença entre a Figura 3.3b e a Figura 3.3a.

Como visto na Figura 3.4, quase não há pixels nos objetos considerados normais na cena em questão, isso ocorre devido a grande similaridade entre a imagem produzida pelo Autoencoder e a imagem original presente no vídeo. Contudo, ao considerar um objeto anômalo, a construção do mesmo não é feita da melhor forma possível, uma vez que o modelo não entende como reproduzi-lo da maneira correta. Dessa forma, a imagem resultante do Autoencoder não terá todos os pixels presente na imagem original, resultando nos pixels exibidos na Figura 3.4.

A imagem que representa o erro é um recurso fundamental para o desenvolvimento da técnica deste trabalho, é através dela que será possível classificar uma imagem como anômala ou normal. Para isso, cada imagem que representa o erro entre uma imagem gerada pelo Autoencoder Convolucional e a imagem original é utilizada em um classificador.

3.3.2 Classificador

Nesta etapa da técnica de classificação automática de anomalias em vídeo, primeiramente, é necessário que as imagens que serão utilizadas durante o treinamento do modelo estejam rotuladas. Para isso, cada imagem que possui algum tipo de anomalia para o contexto que está sendo aplicado é mapeada com um rótulo classificando-a como anomalia. O mesmo ocorre em imagens que não possuem anomalia, porém são rotuladas como imagens normais.

No exemplo da figura 3.5a é possível observar que não há nenhuma anomalia presente no cenário, logo seu rótulo é da classe normal. Já na imagem 3.5b é vista uma anomalia representada pelo veículo, nesse caso a classe do rótulo é definida como anomalia.

(a) Imagem sem anomalia com rótulo da classe normal

(b) Imagem com anomalia com rótulo da classe anomalia

Figura 3.5: Exemplos de imagens que estarão rotuladas para o desenvolvimento da técnica.

Como cada imagem possui um rótulo associado, após o cálculo do erro é possível usá-lo no treinamento do classificador, de modo que o mesmo aprenda a distinguir os erros causados pela ocorrência de anomalia, dos erros obtidos através da perda de dados durante a execução do Autoencoder Convolucional.

Sabendo que o tipo de dado que será utilizado pelo classificador é o erro obtido pela diferença entre a imagem original e a imagem gerada pelo Autoencoder Convolucional, é necessário ter uma base de dados para ser utilizada durante o treinamento do classificador.

A base para o treinamento do classificador é composta por imagens que representam o erro considerando imagens geradas pelo Autoencoder Convolucional e as imagens originais presentes no vídeo. Para a geração do erro é necessário que sejam consideradas apenas imagens que não estavam presentes no conjunto de treinamento do Autoencoder. Dessa forma, o classificador conseguirá aprender a identificar uma imagem com anomalia a partir do erro.

A imagem de erro também possui um rótulo que permite categorizar como anômala ou normal. O classificador utilizará a distribuição dos pixels presentes na imagem de erro para aprender a distinguir um erro mínimo causado por algum tipo de perda de informação durante a execução do Autoencoder de um erro ocasionado pela presença de anomalia.

Após o fim do treinamento do Autoendoder Convolucional e do classificador já é possível obter a classificação de eventos anômalos encontrados nos vídeos. De forma a validar os resultados obtidos, alguns experimentos foram elaborados e executados para quantificar a taxa de sucesso e confiabilidade da técnica proposta no presente trabalho.

Capítulo 4

Avaliação experimental

Neste capítulo serão apresentados os experimentos utilizados para validação da técnica para detecção de anomalias proposta no capítulo 3. Além disso, também será exposto a metodologia utilizada durante a experimentação, bem como a base de dados aplicada. Por fim, será realizado uma análise para verificar o desempenho da técnica de classificação automática de anomalias em vídeo.

4.1 Objetivos dos experimentos

Na seção 3 foi apresentada uma proposta para solucionar o problema de detecção de anomalias em vídeos. Tal proposta consiste em utilizar um modelo Autoencoder Convolucional e um classificador para obter um método capaz de identificar casos de anomalias através do aprendizado das características presentes nas imagens.

Tanto o Autoencoder Convolucional quanto o classificador são modelos que precisam de bastante treinamento para que a qualidade do resultado não seja prejudicada. Assim, faz-se necessário utilizar um plano de treinamento eficiente possibilitando com que ambos os modelos sejam capazes de produzir resultados satisfatórios com diferentes conjuntos de imagens.

Levando em consideração a importância do treinamento dos modelos de rede neural utilizados na proposta deste trabalho, o objetivo dos experimentos é avaliar

cada modelo de forma a garantir a viabilidade do uso para resolução do problema de detecção de anomalias. Tal avaliação será feita com base na análise do desempenho feita a partir de vários testes com um conjunto de dados.

4.2 Metodologia

A técnica de classificação automática de anomalias em vídeos possui duas etapas. A primeira consiste no uso do modelo Autoencoder Convolucional para gerar o aprendizado a cerca dos objetos presentes nas imagens de um vídeo. Nessa etapa, é necessário que o modelo esteja configurado com os hiperparâmetros que consigam fornecer os melhores resultados para serem usados na segunda etapa.

A validação do Autoencoder Convolucional é feita a partir de diversos testes alternando o valor dos parâmetros. Ao final de cada teste, um valor que representa o erro será gerado para que seja possível quantificar a qualidade daquela configuração do modelo. O valor do erro é calculado através da diferença entre a imagem original utilizada como entrada no Autoencoder Convolucional e a imagem gerada pelo mesmo.

Os hiperparâmetros do Autoencoder Convolucional que são alterados durante os testes são: função de ativação na camada de saída, valor da taxa de aprendizado, *batch normalization*, uso da função de ativação *leaky reLU* como função de ativação nas camadas intermediárias e filtros de convolução.

Para a função de ativação na camada de saída foram utilizadas duas em específico: sigmoid e tahn. Essas duas funções de ativação foram escolhidas, pois foi observado que tanto a função sigmoid quanto a tahn são amplamente usadas na literatura.

Seguindo o mesmo motivo do uso das funções de ativação na camada de saída, para os testes, o otimizador Adam foi utilizado contendo um intervalo de variação na taxa de aprendizado. Tal variação é necessária para encontrar o melhor valor que possibilite com que o Autoencoder Convolucional seja treinado de forma mais eficiente, alcançando melhores resultados em poucos ciclos de treinamento.

Visando também a otimização do tempo necessário para o treinamento do Autoencoder Convolucional, os testes foram executados utilizando o mecanismo denominado *batch normalization*. Este mecanismo é capaz de corrigir as variações obtidas na entrada de cada camada convolucional e também possui um efeito benéfico no cálculo do gradiente, reduzindo as dependências dos seus valores iniciais. (IOFFE; SZEGEDY, 2015)

Os filtros de convolução têm o objetivo de capturar as características principais da imagem que está sendo processada pelas camadas convolucionais. Durante os testes, a quantidade de filtros utilizada é variada em um conjunto de valores com o objetivo de obter o melhor resultado.

Além do Autoencoder Convolucional, o classificador também possui alguns dos seus hiperparâmetros alternáveis durante o experimento, onde estes são: taxa de aprendizado, uso de batch normalization, uso da função de ativação Leaky ReLU ou ReLU, filtros de convolução e taxa de neurônios que são desativados em cada camada do classificador.

Assim como no Autoencoder Convolucional, o otimizador utilizado na taxa de aprendizado foi o Adam. Os hiperparâmetros batch normalization, função de ativação e filtros de convolução possuem os mesmos conceitos dos que foram apresentados para o Autoencoder Convolucional. A diferença nesse caso está na taxa de neurônios que são desativados em cada camada do modelo.

A taxa de neurônio que é desativada nas camadas do classificador é conhecida como *Dropout*, onde está é conceituada como uma técnica que randomicamente seleciona alguns neurônios e os ignora durante o treinamento. Isto ocorre para que a rede neural não fique muito específica, fazendo com que se torne frágil demais a pequenas mudanças. (SRIVASTAVA et al., 2014)

Baseado nas alterações dos valores dos hiperparâmetros, foram desenvolvidos dois experimentos. O primeiro consiste em avaliar o Autoencoder Convolucional usando um modelo classificador já treinado. O segundo experimento consiste em treinar o classificador, onde dessa vez foi utilizado um modelo Autoencoder Convolucional

após a fase de treinamento. Os experimentos serão referenciados neste trabalho como experimento I e II a partir deste ponto.

Uma vez que os experimentos foram elaborados, é necessário uma base de dados que seja fiel à realidade para que seja possível simular o uso da técnica de detecção de anomalias com situações do cotidiano. Pensando nisso, foi utilizada uma base de dados popular entre os trabalhos relacionado à detecção de anomalias, onde esta será abordada a seguir.

4.3 Base de dados

Como o treinamento do modelo para classificação de anomalias em vídeo necessita de um conjunto de dados, para este trabalho, foi utilizada uma base de dados disponibilizada por Mahadevan et al. (2010). Nesta base de dados, é possível encontrar dois subconjuntos de dados, Peds1 e Peds2, contendo diferentes vídeos retratando a realidade de um determinado contexto de modo que seja usado para detecção de anomalias.

O subconjunto de dados Peds1 contém 70 vídeos, sendo 32 contendo somente eventos normais que devem ser utilizados para o treinamento do modelo Autoencoder Convolucional e classificador, e 36 vídeos com anomalias para validação e análise do desempenho dos modelos.

Já o subconjunto de dados Peds2 é um pouco menor contendo 28 vídeos no total, sendo 16 para o treinamento e 12 para os testes. O subconjunto de dados Peds2 é formado por vídeos capturados em locais diferentes do Peds1, porém os critérios de anomalia são iguais em ambos subconjuntos de dados.

Tanto nos vídeos do Peds1 quanto no Peds2 é possível perceber que a frequência de pedestres varia bastante. Há momentos em que muitos pedestres estão em cena e outros contendo poucos. Considerando o Peds1 a Figura 4.1 exibe dois exemplos do caso onde é possível observar vários pedestres caminhando, já na Figura 4.2, ocorre o contrário.

Figura 4.1: Exemplos de imagens onde é possível encontrar muitas pessoas em cena

Figura 4.2: Exemplos de imagens onde é possível encontrar poucas pessoas em cena

Nessa base de dados, a ocorrência de anomalias incluem: ciclistas, skatistas, carros e pessoas andando pela grama. Também há vídeos que contém um ou mais cadeirantes, nesse caso também é considerado como uma anomalia. A Figura 4.3 exibe alguns exemplos de anomalias que podem ser encontradas nos vídeos presentes em Peds1.

As anomalias ocorrem em diferentes momentos nos vídeos, podendo ser quando há muitos pedestres no caminho ou quando a quantidade é mínima. Isso permite com que a base tenha maior qualidade ao retratar a realidade daquele ambiente, permitindo com que a técnica de detecção de anomalia consiga abranger a maior quantidade de situações possíveis.

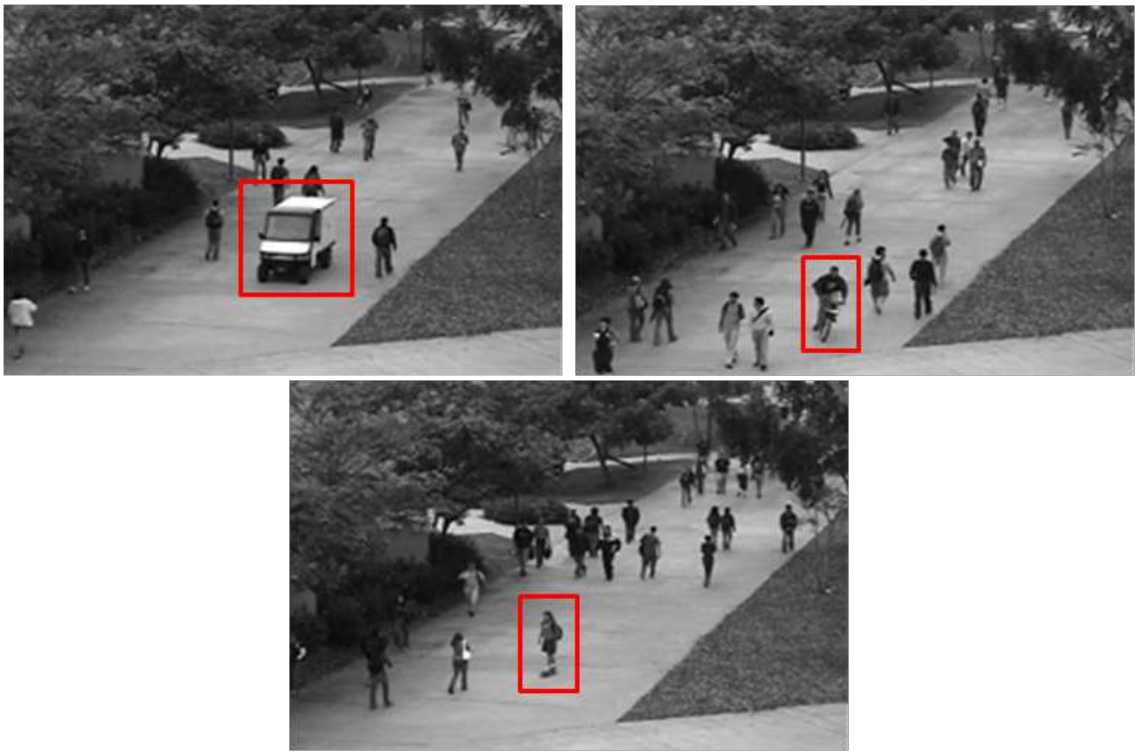


Figura 4.3: Exemplos de anomalias encontradas na base de dados Peds1.

4.4 Métrica de Avaliação

Uma vez que a técnica tenha sido desenvolvida e os experimentos elaborados, é necessário uma forma de validar os resultados obtidos ao final. Para isso, foi preciso utilizar uma métrica que consiga fornecer um valor que pode ser utilizado para medir a qualidade da técnica proposta.

A métrica utilizada para análise do resultado é a acurácia. Esta métrica, no geral, mede a proporção de previsões corretas sobre o número total de instâncias avaliadas. Dessa forma, após a realização do experimento é possível observar de maneira clara e objetiva a real qualidade da técnica proposta neste trabalho. (HOSSIN; SULAIMAN, 2015)

Considerando uma abordagem em duas etapas para a técnica deste trabalho, Autoencoder Convolucional e classificador, a acurácia é utilizada no resultado fornecido pelo classificador. Isto ocorre pois o classificador é o algoritmo que de fato acusa presença de anomalia ou não considerando as imagens de um vídeo.

4.5 Análise dos Experimentos

Partindo do ponto em que já há uma forma de avaliação da técnica apresentada nesse trabalho, a seguir será exibido a estrutura dos modelos utilizados, Autoencoder Convolucional e Classificador.

4.5.1 Classificador

Sabendo que o objetivo do classificador é dizer de fato se há ou não alguma anomalia em um vídeo, é necessário que o modelo seja capaz de aprender as características que compõem uma anomalia. Como dito na seção 3, o classificador receberá como dado de entrada uma imagem que representa a diferença entre a imagem original e a que foi reproduzida pelo Autoencoder Convolucional.

A partir deste ponto, todas as características de definem uma anomalia para o contexto aplicado serão compreendidas pelo modelo, permitindo com que um resultado de boa qualidade seja gerado. A estrutura do classificador utilizado será explicitada se acordo com a tabela 4.1.

Classificador	1ª camada	2ª camada	3ª camada
Filtros	16	32	64
Matriz de filtros	3x3	3x3	3x3
Stride	2,2	2,2	2,2
BatchNormalization	não usado	não usado	não usado
Função de ativação	LeakyReLU	LeakyReLU	LeakyReLU
Dropout	0.1	0.1	0.1

Tabela 4.1: Hiperparâmetros utilizados no classificador

A configuração de classificador exibida na tabela 4.1 representa o classificador utilizado no Experimento I, visto que este experimento visa apenas otimizar os hiper-parâmetros do Autoencoder.

4.5.2 Autoencoder Convolucional

De modo a facilitar o entendimento, o modelo Autoencoder Convolucional será explicitado em duas partes: codificação e decodificação.

Na etapa codificação foram utilizadas três camadas convolucionais, onde a primeira possui 128 filtros, a segunda 256 e a terceira 256. Após a terceira camada é esperado que as principais características da imagens já tenha sido capturado pelos filtros, permitindo com que seja possível refazer a imagem novamente na etapa decodificação.

Além dos filtros foi definido o tamanho da matriz que representa um filtro. Na primeira e segunda camada convolucional da etapa codificação foi utilizado um tamanho 3x3 para a matriz do filtro e na terceira o tamanho 5x5. Complementando o uso do filtro, foi utilizado o valor 2 para o stride, tanto para a horizontal quanto para a vertical, exceto na terceira camada convolucional, onde dessa vez o valor utilizado para o stride foi 5.

Na etapa decodificação, também foram utilizadas três camadas convolucionais, porém a quantidade de filtros em cada camada convolucional é ligeiramente diferente se comparada ao que foi dito na codificação A primeira camada convolucional da etapa decodificação possui 256 filtros, a segunda 128 e a terceira e última apenas 1.

O tamanho da matriz de filtro na etapa decodificação segue o mesmo da codificação, assim como o valor usado para o stride. É importante ter em mente o objetivo de cada etapa para obter a total compreensão do modelo. O codificador tem o papel de aprender as características da imagem e o decodificador, por sua vez, tem o objetivo de reproduzir a imagem a partir da codificação gerada com base nas características aprendidas. É possível observar a estrutura utilizada para o Autoencoder Convolucional na tabela 4.2 e 4.3.

Codificador	1ª camada	2ª camada	3ª camada
Filtros	128	256	256
Matriz de filtros	3x3	3x3	5x5
Stride	2,2	2,2	5,5
BatchNormalization	não usado	não usado	não usado
Função de ativação	LeakyReLU	LeakyReLU	LeakyReLU

Tabela 4.2: Hiperparâmetros utilizados na parte de codificação do Autoencoder Convolucional

Como observado nas tabelas 4.2 e 4.3, é possível perceber que o decodificador possui um comportamento que se semelhante à ordem contrária a disposição de

Decodificador	1ª camada	2ª camada	3ª camada
Filtros	256	128	1
Matriz de filtros	5x5	3x3	3x3
Stride	5,5	2,2	5,5
BatchNormalization	não usado	não usado	não usado
Função de ativação	LeakyReLU	LeakyReLU	tanh

Tabela 4.3: Hiperparâmetros utilizados na parte de decodificação do Autoencoder Convolucional

camadas do codificador. Isso ocorre pois é esperado que o modelo consiga produzir a imagem que está codificada de forma que a diferença se comparada com a imagem original seja mínima.

A configuração de Autoencoder exibida nas tabelas 4.2 e 4.3 representa o Autoencoder utilizado no Experimento II, visto que este experimento visa apenas otimizar os hiper-parâmetros do Autoencoder. Essa configuração também representa os hiper-parâmetros para o Autoencoder que resultaram na maior acurácia durante o Experimento I.

4.5.3 Grid Search

Os experimentos fazem uso de uma técnica de busca de hiperparâmetros ótimos exaustiva chamada *Grid Search*. Essa técnica consiste na avaliação do modelo repetidamente para cada combinação de valores dos hiperparâmetros escolhidos. Após avaliar o modelo com todos os valores de parâmetros, a iteração que rendeu a melhor métrica de avaliação é encontrada.

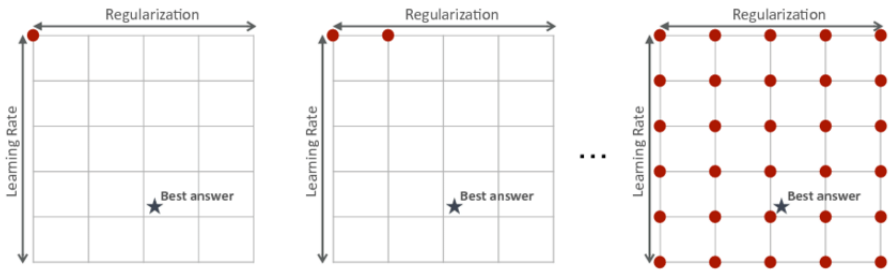


Figura 4.4: Técnica de *Grid Search*, onde cada ponto vermelho simboliza uma avaliação do modelo utilizando uma combinação de dois parâmetros.

4.5.4 Experimento I

O primeiro experimento possui o objetivo de validar o uso do Autoencoder Convolucional de forma que seja possível analisar os resultados obtidos considerando diferentes situações de anomalias presentes na base de dados utilizada. Como nesse ponto o classificador já se apresenta treinado, é esperado que ele esteja apto a classificar de maneira correta a ocorrência de anomalias.

Os testes nesta etapa basearam-se na busca pelos melhores parâmetros do Autoencoder Convolucional. Para isso, foi utilizado a técnica *Grid Search*, onde este, por sua vez, é uma otimização dos hiperparâmetros de uma rede neural através de várias tentativas seguidas. A configuração que conseguir gerar os melhores resultados será considerada qualificada para o uso no modelo. (LIASHCHYNSKYI; LIASHCHYNSKYI, 2019)

O experimento consiste no treinamento de um Autoencoder com as configurações baseadas no Autoencoder representado pelas tabelas 4.2 e 4.3, porém com os seguintes parâmetros variando:

Parâmetros	**Valores**
Tam. Camada de Compressão	32, 64, 128, 256
Leaky ReLU	usado, não usado
Batch Normalization	usado, não usado
Taxa de Aprendizado	0.0005, 0.00005
Função de ativação output	sigmoid, tanh

Tabela 4.4: Hiperparâmetros do Autoencoder a serem otimizados no Experimento I

O parâmetro de Tamanho de Camada de Compressão indica o tamanho da camada mais interna do autoencoder. Já o parâmetro Leaky ReLU indica se as camadas intermediárias do Autoencoder utilizam um ReLU Leaky ou um ReLU normal. O parâmetro Batch Normalization indica se após cada camada do Autoencoder os valores de input para a próxima camada devem ser normalizados baseando-se no lote atual. A Função de Ativação no Output pode variar entre os algoritmos de Sigmoide e Tangente Hiperbólica.

Ao fim do experimento os parâmetros serão escolhidos baseados na acurácia do Classificador descrito na tabela 4.1. Cada iteração do *Grid Search* treina o Autoen-

coder por no máximo 50 épocas (com critério de parada por paciência baseado no valor de perda de validação), com gradiente descendente em lotes de 32. O dataset é dividido em imagens anômalas e não-anômalas, onde uma quantidade igual de imagens anômalas e não-anômalas é usado como base para treino, teste e validação, ou seja, o autoencoder é treinado e testado com uma proporção de 50% de imagens anômalas. Após o treino do Autoencoder, o Classificador é treinado com as imagens de output do Autoencoder e então as classifica, dando sua acurácia.

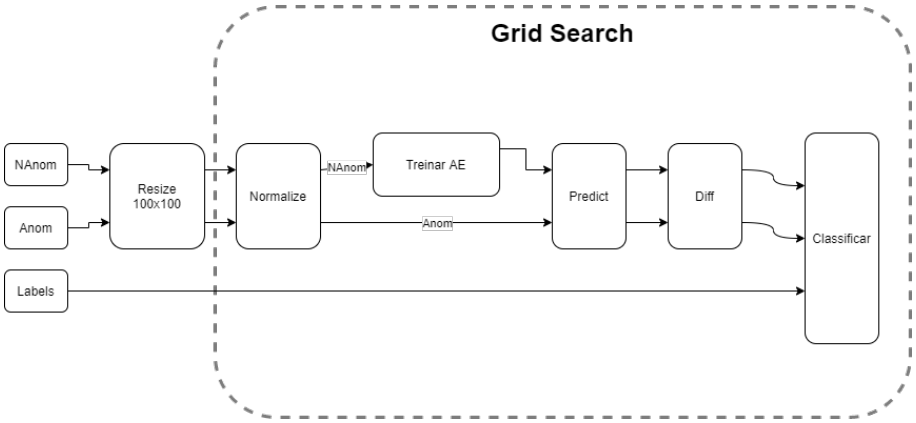


Figura 4.5: Fluxo do Experimento I.

4.5.5 Experimento II

O segundo experimento tem o intuito de obter a melhor configuração para o classificador. Nesta etapa, já é utilizado um Autoencoder Convolucional treinado (com a melhor configuração encontrada no Experimento I), fazendo com que a avaliação seja feita apenas com o classificador. Da mesma forma que o Experimento I, o algoritmo *Grid Search* também foi utilizado, permitindo uma vasta gama de testes com diferentes valores para os hiperparâmetros do modelo.

Ao final de cada experimento é possível obter uma visão geral dos resultados obtidos, o que permite com que a melhor configuração seja escolhida. Dessa forma, como a ocorrência de anomalia pode ser dinâmica em alguns casos, obter a melhor configuração considerando uma base de dados com diferentes exemplos permite com que a maior quantidade de situações seja detectada pela técnica.

O experimento consiste no treinamento de um Classificador com as configurações

baseadas no Classificador representado pela tabela 4.1, porém com os seguintes parâmetros variando:

Parâmetros	Valores
Tam. Camadas	{16,32,64}, {32,64,128}, {64,128,256}, {128,256,512}
Leaky ReLU	usado, não usado
Batch Normalization	usado, não usado
Taxa de Aprendizado	0.001, 0.0001, 0.00001
Taxa de Dropout	0.05, 0.1, 0.15, 0.2, 0.25

Tabela 4.5: Hiperparâmetros do Classificador a serem otimizados no Experimento II

O parâmetro de Tamanho das Camadas indica a quantidade de filtros utilizadas nas três camadas do Classificador a cada iteração. Já o parâmetro Leaky ReLU indica se as camadas intermediárias do Classificador utilizam um ReLU Leaky ou um ReLU normal. O parâmetro Batch Normalization indica se após cada camada do Classificador os valores de input para a próxima camada devem ser normalizados baseando-se no lote atual. O parâmetro de Taxa de Dropout indica a porcentagem de neurônios em cada camada que devem ser inutilizados para evitar overfitting.

Ao fim do experimento os parâmetros serão escolhidos baseados na acurácia do próprio Classificador recém treinado. Cada iteração do *Grid Search* treina o Classificador por no máximo 50 épocas (com critério de parada por paciência baseado no valor de perda de validação), com gradiente descendente em lotes de 32. O dataset é dividido em imagens anômalas e não-anômalas, onde uma quantidade igual de imagens anômalas e não-anômalas é usado como base para treino, teste e validação, ou seja, o Classificador e o Autoencoder são treinados e testados com uma proporção de 50% de imagens anômalas.

4.6 Resultados

Nesta seção serão apresentados os resultados encontrados após a execução sequencial dos dois Experimentos descritos nos capítulos anteriores. Para demonstrar a relação de cada hiperparâmetro com o resultado (acurácia), serão exibidos mapas de calor para cada par de hiperparâmetros. Os valores dos mapas de calor representam a acurácia do classificador ao final de cada iteração do experimento e são agregados

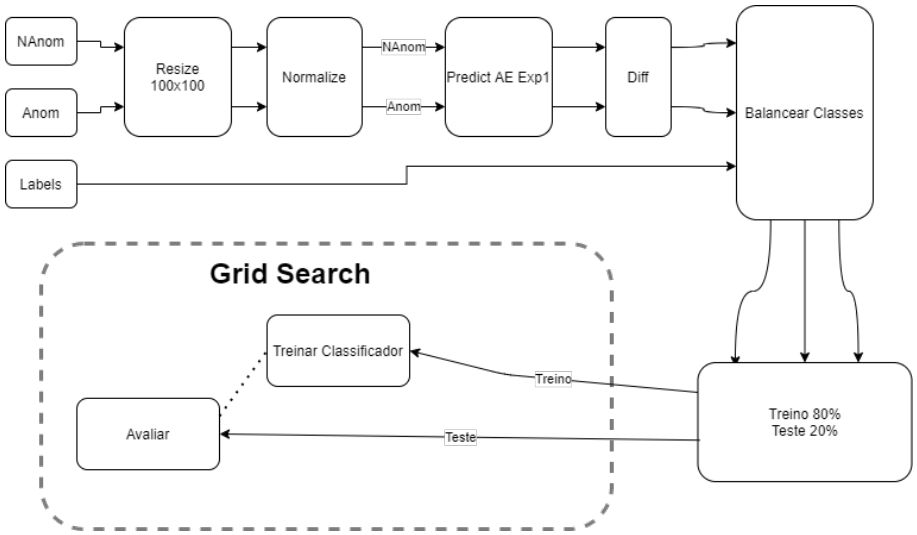


Figura 4.6: Fluxo do Experimento II.

por média para o caso de pares de parâmetros iguais.

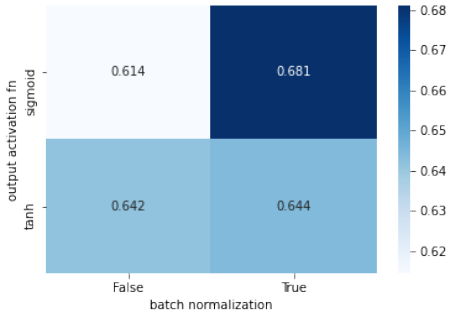
4.6.1 Experimento I

O Experimento I tem como objetivo otimizar os resultados da rede ao buscar os melhores hiper-parâmetros para o Autoencoder Convolucional através da técnica de *Grid Search*, utilizando como métrica de avaliação a acurácia de um Classificador Convolucional com hiper-parâmetros já preestabelecidos.
Os hiper-parâmetros do Autoencoder a serem otimizados são:

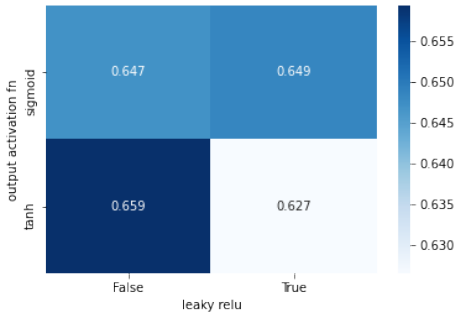
- Função de Ativação na Camada de Saída
- Taxa de Aprendizado
- Normalização em Lotes
- Leaky ReLU
- Tamanho da Camada de Compressão

Cada um destes parâmetros serão discutidos nas subseções a seguir.

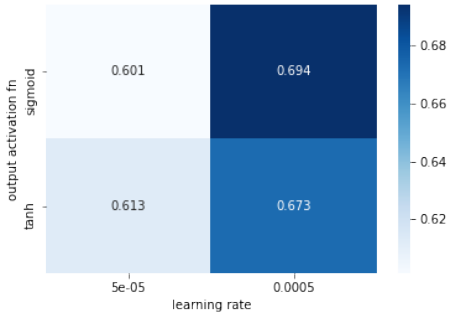
4.6.1.1 Função de Ativação na Camada de Saída



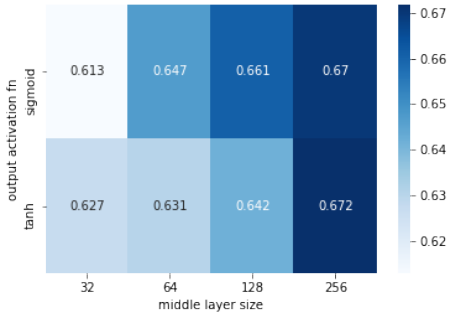
(a) Função de Ativação × Normalização em Lotes



(b) Função de Ativação × Leaky ReLU



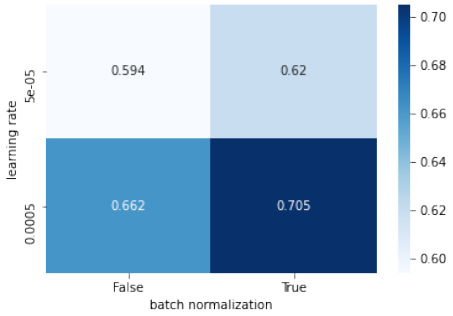
(c) Função de Ativação × Taxa de Aprendizado



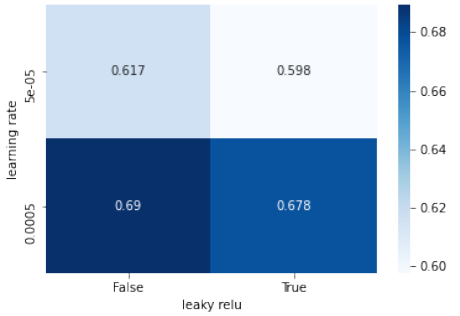
(d) Função de Ativação × Tamanho da Camada de Compressão

Figura 4.7: Mapas de Calor para o parâmetro de Função de Ativação na Camada de Saída

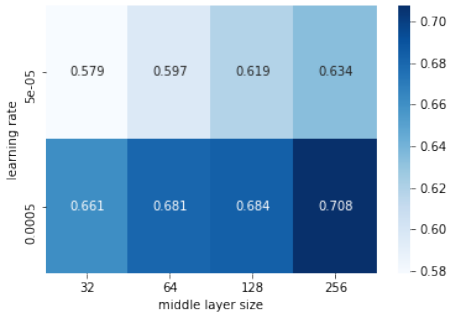
4.6.1.2 Taxa de Aprendizado



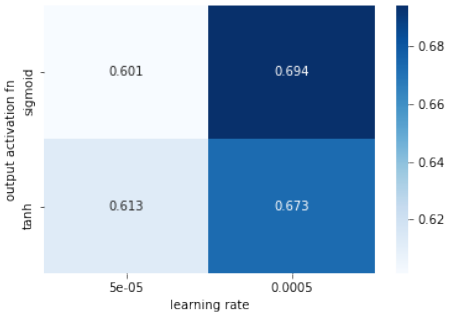
(a) Taxa de Aprendizado × Normalização em Lotes



(b) Taxa de Aprendizado × Leaky ReLU



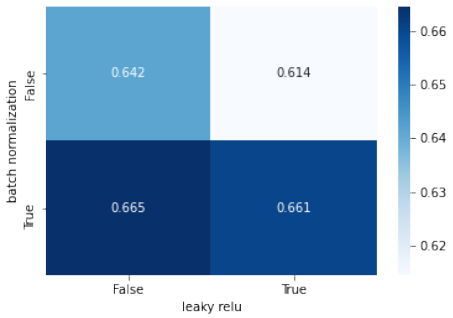
(c) Taxa de Aprendizado × Tamanho da Camada de Compressão



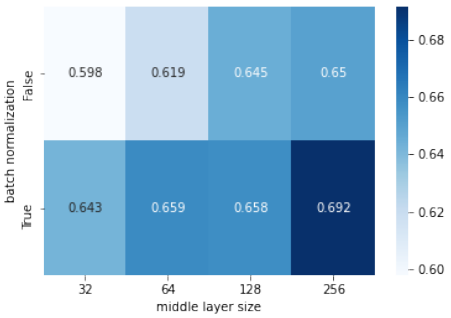
(d) Taxa de Aprendizado × Função de Ativação

Figura 4.8: Mapas de Calor para o parâmetro de Taxa de Aprendizado

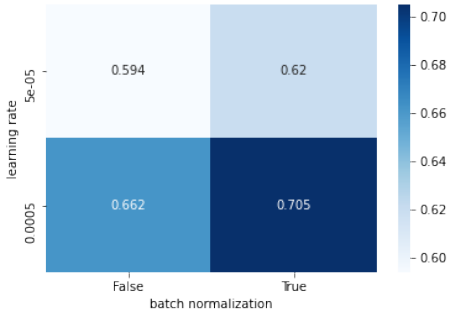
4.6.1.3 Normalização em Lotes



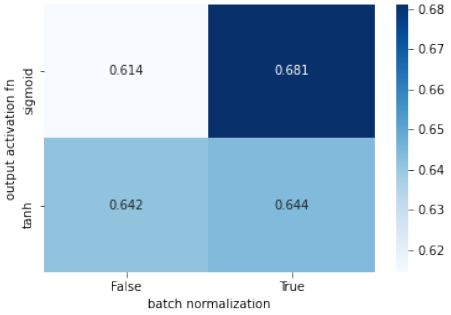
(a) Normalização em Lotes × Leaky ReLU



(b) Normalização em Lotes × Tamanho da Camada de Compressão



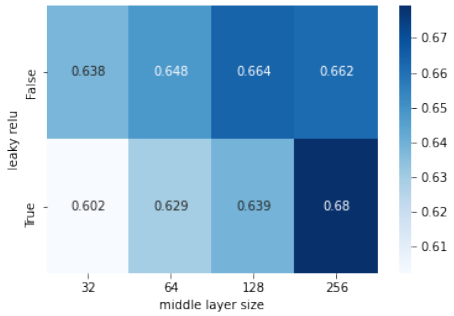
(c) Normalização em Lotes × Taxa de Aprendizado



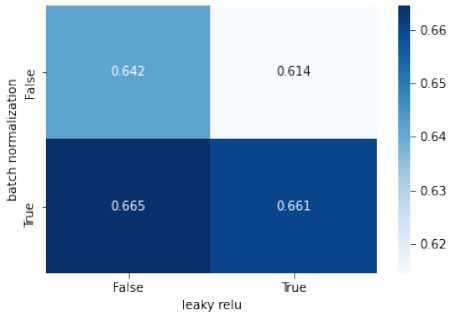
(d) Normalização em Lotes × Função de Ativação

Figura 4.9: Mapas de Calor para o parâmetro de Normalização em Lotes

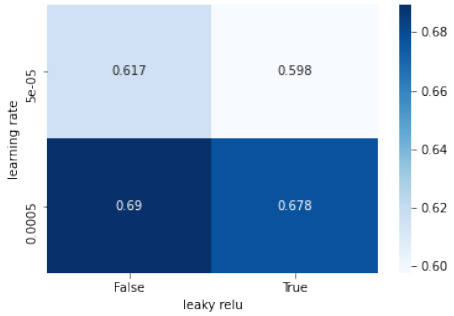
4.6.1.4 Leaky ReLU



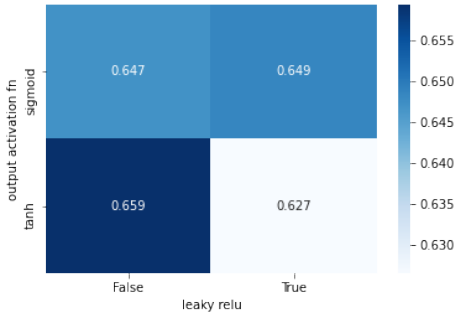
(a) Leaky ReLU × Tamanho da Camada de Compressão



(b) Leaky ReLU × Normalização em Lotes



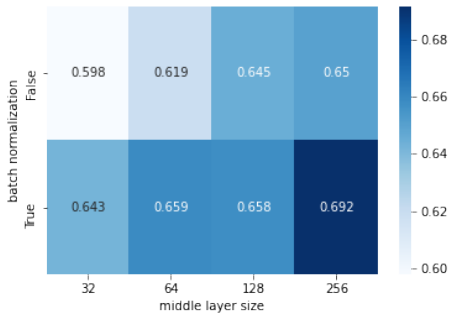
(c) Leaky ReLU × Taxa de Aprendizado



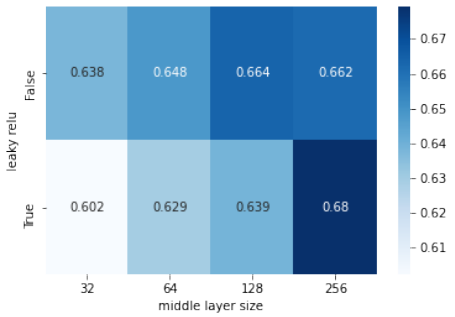
(d) Leaky ReLU × Função de Ativação

Figura 4.10: Mapas de Calor para o parâmetro de Leaky ReLU

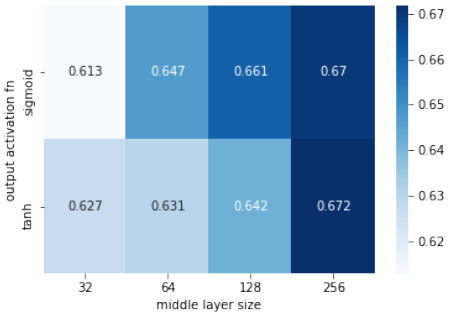
4.6.1.5 Tamanho da Camada de Compressão



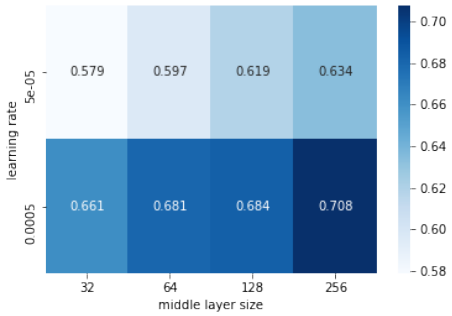
(a) Tamanho da Camada de Compressão × Normalização em Lotes



(b) Tamanho da Camada de Compressão × Leaky ReLU



(c) Tamanho da Camada de Compressão × Função de Ativação



(d) Tamanho da Camada de Compressão × Taxa de Aprendizado

Figura 4.11: Mapas de Calor para o parâmetro de Tamanho da Camada de Compressão

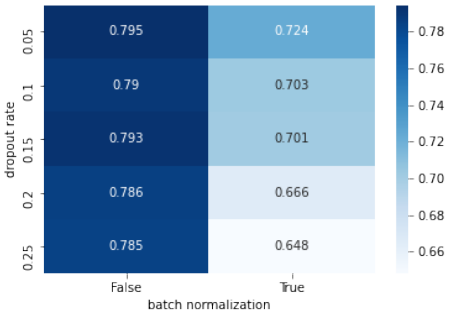
4.6.2 Experimento II

Este experimento tem como objetivo otimizar os resultados da rede ao buscar os melhores hiper-parâmetros para o Classificador através da técnica de *Grid Search*, utilizando como métrica de avaliação a acurácia do próprio Classificador e utilizando um Autoencoder com os melhores hiperparâmetros encontrados pelo Experimento I.. Os hiper-parâmetros do Classificador a serem otimizados são:

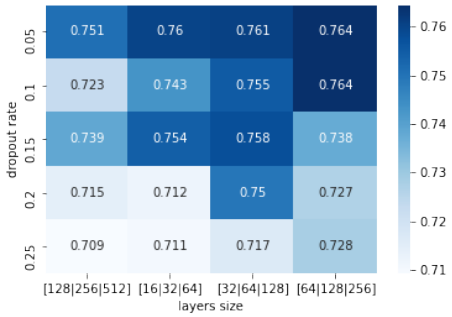
- Taxa de Dropout
- Taxa de Aprendizado
- Normalização em Lotes
- Leaky ReLU
- Tamanho das Camadas

Cada um destes parâmetros serão discutidos nas subseções a seguir.

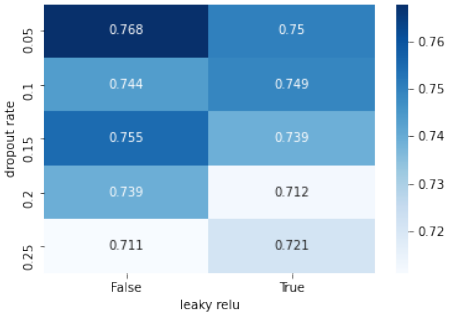
4.6.2.1 Taxa de Dropout



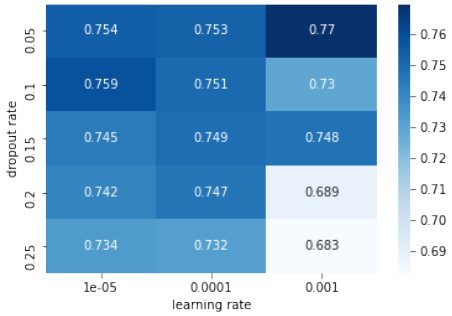
(a) Taxa de Dropout × Normalização em Lotes



(b) Taxa de Dropout × Tamanhos das Camadas



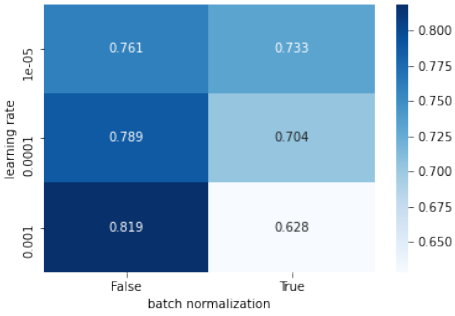
(c) Taxa de Dropout × Leaky ReLU



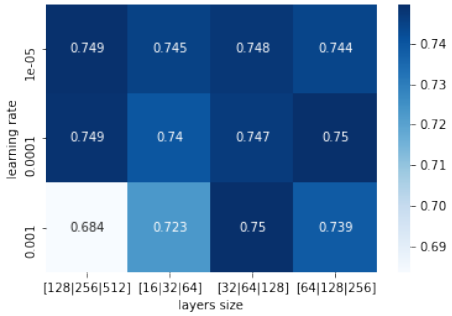
(d) Taxa de Dropout × Taxa de Aprendizado

Figura 4.12: Mapas de Calor para o parâmetro de Taxa de Dropout

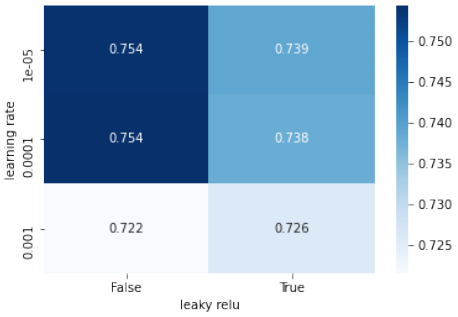
4.6.2.2 Taxa de Aprendizado



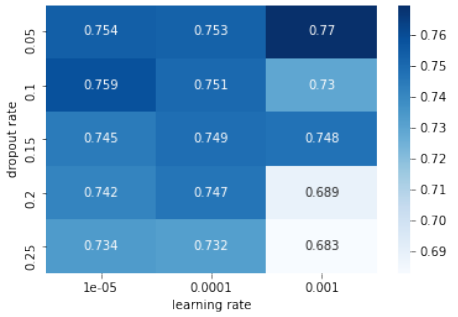
(a) Taxa de Aprendizado × Normalização em Lotes



(b) Taxa de Aprendizado × Tamanhos das Camadas



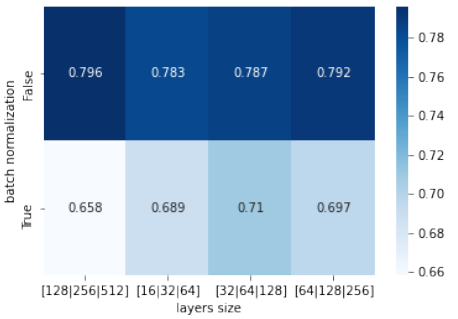
(c) Taxa de Aprendizado × Leaky ReLU



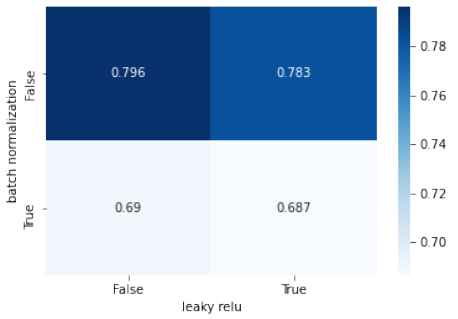
(d) Taxa de Aprendizado × Taxa de Dropout

Figura 4.13: Mapas de Calor para o parâmetro de Taxa de Aprendizado

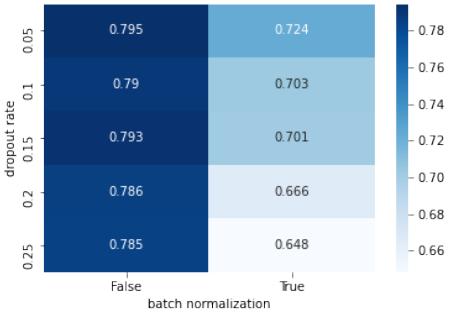
4.6.2.3 Normalização em Lotes



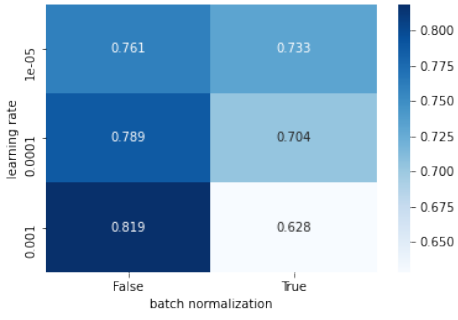
(a) Normalização em Lotes × Tamanhos das Camadas



(b) Normalização em Lotes × Leaky ReLU



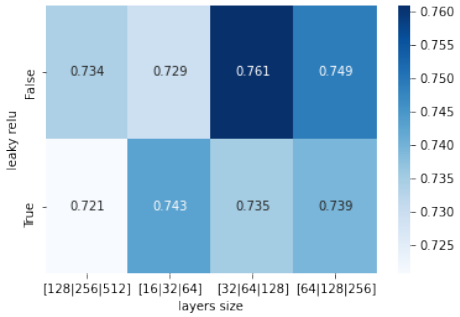
(c) Normalização em Lotes × Taxa de Dropout



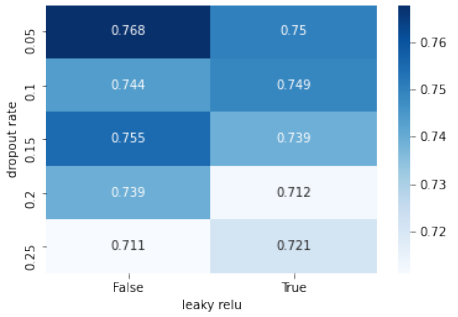
(d) Normalização em Lotes × Taxa de Aprendizado

Figura 4.14: Mapas de Calor para o parâmetro de Normalização em Lotes

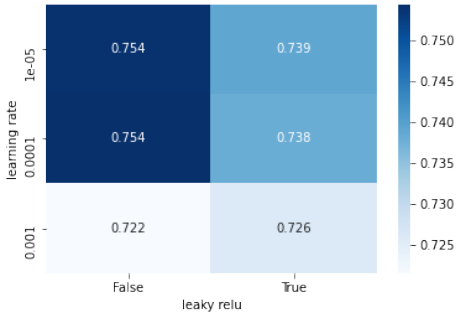
4.6.2.4 *Leaky ReLU*



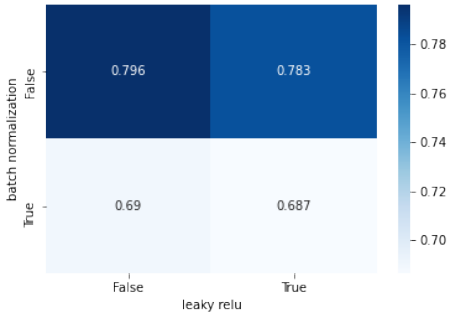
(a) Leaky ReLU × Tamanhos das Camadas



(b) Leaky ReLU × Taxa de Dropout



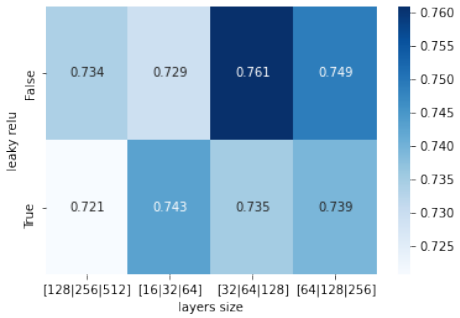
(c) Leaky ReLU × Taxa de Aprendizado



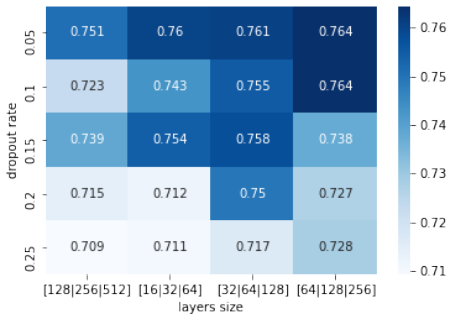
(d) Leaky ReLU × Normalização em Lotes

Figura 4.15: Mapas de Calor para o parâmetro de Leaky ReLU

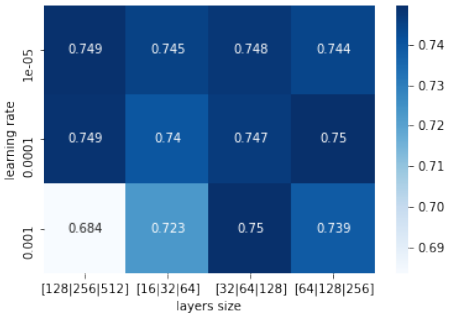
4.6.2.5 *Tamanho das Camadas*



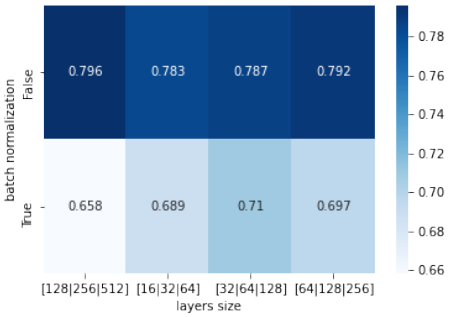
(a) Tamanho das Camadas × Leaky ReLU



(b) Tamanho das Camadas × Taxa de Dropout



(c) Tamanho das Camadas × Taxa de Aprendizado



(d) Tamanho das Camadas × Normalização em Lotes

Figura 4.16: Mapas de Calor para o parâmetro de Tamanho das Camadas

4.6.3 Conclusões

Para o experimento I, os hiperparâmetros que resultaram na maior acurácia foram:

Parâmetros	**Valor**
Tam. Camada de Compressão	256
Leaky ReLU	usado
Batch Normalization	usado
Taxa de Aprendizado	0.0005
Função de ativação output	sigmoid

Tabela 4.6: Melhores Hiperparâmetros do Autoencoder encontrados no Experimento I

O Autoencoder com os hiperparâmetros da tabela 4.6 resultou em uma acurácia de 77.62%.

Para o experimento II, os hiperparâmetros que resultaram na maior acurácia foram:

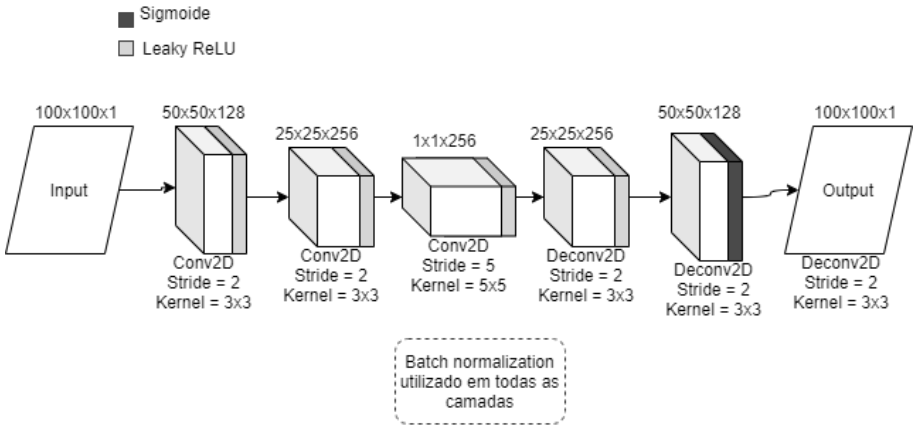


Figura 4.17: Configuração do AutoEncoder com os melhores parâmetros.

Parâmetros	**Valores**
Tam. Camadas	{128,256,512}
Leaky ReLU	não usado
Batch Normalization	usado
Taxa de Aprendizado	0.001
Taxa de Dropout	0.15

Tabela 4.7: Melhores Hiperparâmetros do Classificador encontrados no Experimento II

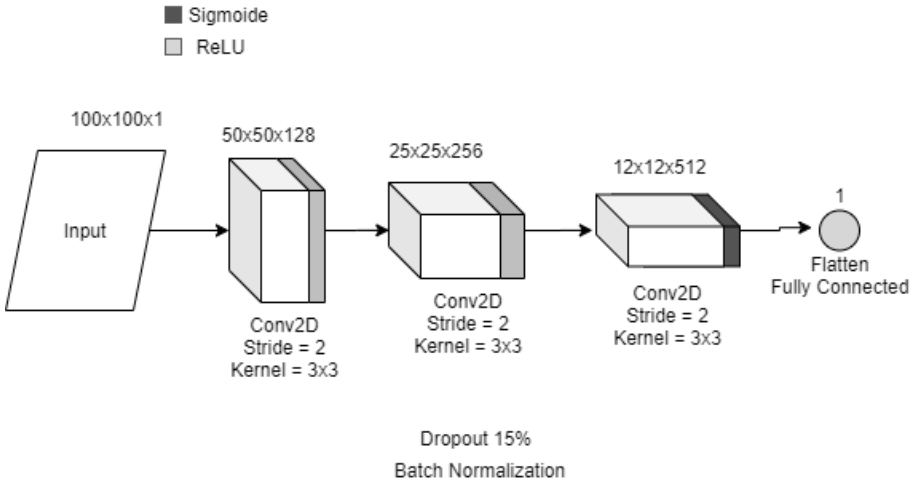


Figura 4.18: Configuração do Classificador com os melhores parâmetros.

O Classificador com os hiperparâmetros da tabela 4.7 resultou em uma acurácia de 87.44%.

Capítulo 5

Conclusões

Neste capítulo serão apresentadas as considerações finais obtidas acerca do trabalho proposto, suas contribuições, bem como possíveis próximos passos para dar seguimento na evolução da técnica.

5.1 Considerações finais sobre o trabalho

Este trabalho busca desenvolver uma técnica para detecção de anomalias em vídeos baseada no aprendizado das características presentes em cada imagem. O modelo proposto é definido por um Autoencoder que irá tentar aprender todos os aspectos importantes dos objetos que compõem uma cena de um vídeo. Através disso, será possível identificar os objetos que apresentarem maior erro em relação à imagem original, onde nesse caso serão as anomalias.

A detecção de anomalias é um problema considerado complexo devido as inúmeras possibilidades de como uma anomalia pode acontecer. Considerando a dificuldade em obter um método completamente eficiente, uma vez que o uso de estratégias triviais não permitirá grande abrangência de situações, faz-se necessário a busca por técnicas que consigam identificar ocorrência de anomalia de modo que sua aplicabilidade abranja a maior quantidade de casos possíveis.

Com isso, nesse trabalho foi proposta uma técnica de classificação automática de

anomalias em vídeo através de dois modelos, combinando o aprendizado supervisionado com o aprendizado não supervisionado. Com esta técnica é possível ter alta porcentagem de precisão considerando os resultados obtidos. De modo a comprovar o método, foram elaborados e executados experimentos que possibilitaram a análise dos resultados e avaliação da qualidade dos mesmos.

Os experimentos executados foram capazes de gerar resultados satisfatórios considerando o uso da acurácia como métrica principal. Foi possível observar que a técnica é capaz de alcançar 92% de acurácia para o conjunto de dados utilizados durante os testes executados no experimento.

Com esse resultado é possível afirmar que a estratégia de utilizar duas ferramentas, Autoencoder Convolucional e classificador, combinadas para a resolução do problema é promissora e pode ser melhorada permitindo resultados ainda mais satisfatórios, independente do conjunto de imagens que estiver sendo utilizado.

5.2 Trabalhos futuros

Apesar dos resultados aceitáveis obtidos durante os experimentos, assim como qualquer outra técnica elaborada, é possível estabelecer melhorias e ampliar os espaço de aplicações da técnica para detecção de anomalias presente neste trabalho. Nesta seção serão apresentadas potenciais evoluções para a técnica de detecção de anomalias permitindo com que o resultado possua qualidade superior.

Mesmo com os bons resultados obtidos durante os experimentos, foi identificada uma dificuldade para identificar anomalias considerando os casos onde os objetos anômalos tinham bastante similaridade com os objetos considerados normais na cena. Para a base de dados utilizada, esses casos ocorreram quando pedestres estavam utilizando bicicletas ou skates.

Uma trabalho futuro seria estudar uma forma de identificar anomalias considerando também as características temporais além das espaciais através do uso de redes neurais artificiais recorrentes. Assim, a técnica conseguirá distinguir com mais clareza, por exemplo, uma pessoa caminhando de uma pessoa andando de skate.

Outro desafio interessante é utilizar a técnica de detecção de anomalias em outras bases de dados com diferentes contextos. Dessa forma, seria possível entender melhor como os modelos Autoencoder Convolucional e classificador se comportam e tentar definir, de maneira mais eficiente, a configuração dos hiperparâmetros fazendo com que a maior quantidade de situações seja abrangida.

Referências

ANTHONY, M.; BARTLETT, P. L. *Neural Network Learning: Theoretical Foundations*. 1st. ed. USA: Cambridge University Press, 2009. ISBN 052111862X.

AYYADEVARA, V. *Neural Networks with Keras Cookbook*. Packt Publishing, 2019. ISBN 9781789346640. Disponível em: <https://books.google.com.br/books?id=UbqXwgEACAAJ>.

Braida do Carmo, F. C. Transformando o Problema de Filtragem Colaborativa em Aprendizado de Máquina Supervisionado. 2012. Disponível em: <http://www.cos.ufrj.br/index.php?option=com/publicacao&task=visualizar&id=2361>.

BRUCE, R. A bayesian approach to semi-supervised learning. 02 2002.

DATTA, A.; SHAH, M.; LOBO, N. Person-on-person violence detection in video data. In: . [S.l.: s.n.], 2002. v. 1, p. 433– 438 vol.1. ISBN 0-7695-1695-X.

DEY, N. et al. A survey of image classification methods and techniques. In: . [S.l.: s.n.], 2014.

FEDERAL, U. et al. Luiz Gustavo Hafemann an Analysis of Deep Neural Networks for an Analysis of Deep Neural Networks for. 2014.

GAO, Y. et al. Violence detection using oriented violent flows. *Image and Vision Computing*, v. 48-49, 02 2016.

GONZALEZ, B. *Aprendizado Automático de Características Utilizando Autoencoders Esparsos*. Tese (Doutorado), 2014.

HASAN, M. et al. Learning temporal regularity in video sequences. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 733–742, 2016.

HOSSIN, M.; SULAIMAN, M. A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining & Knowledge Management Process*, Academy & Industry Research Collaboration Center (AIRCC), v. 5, n. 2, p. 1, 2015.

IOFFE, S.; SZEGEDY, C. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015.

JACOBS, R. A. Increased rates of convergence through learning rate adaptation. *Neural Networks*, Pergamon, v. 1, n. 4, p. 295–307, jan 1988. ISSN 08936080. Disponível em: <https://www.sciencedirect.com/science/article/abs/pii/0893608088900032>.

KOOIJ, J. et al. Multi-modal human aggression detection. *Computer Vision and Image Understanding*, v. 144, 01 2015.

KUMAR, V. Parallel and distributed computing for cybersecurity. *IEEE Distributed Systems Online*, v. 6, n. 10, p. 1–9, 2005. ISSN 15414922.

LECUN, Y. et al. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, MIT Press Journals, v. 1, n. 4, p. 541–551, 1989. ISSN 0899-7667.

LIASHCHYNSKYI, P.; LIASHCHYNSKYI, P. Grid search, random search, genetic algorithm: A big comparison for nas. *ArXiv*, abs/1912.06059, 2019.

MAHADEVAN, V. et al. Anomaly detection in crowded scenes. In: IEEE. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. [S.l.], 2010. p. 1975–1981.

MASCI, J. et al. Stacked convolutional auto-encoders for hierarchical feature extraction. In: HONKELA, T. et al. (Ed.). *Artificial Neural Networks and Machine Learning – ICANN 2011*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 52–59. ISBN 978-3-642-21735-7.

MITCHELL, T. M. *Machine Learning*. New York: McGraw-Hill, 1997. ISBN 978-0-07-042807-2.

MOHAMMADI, S. et al. Angry crowds: Detecting violent events in videos. In: . [S.l.: s.n.], 2016. v. 9911, p. 3–18. ISBN 978-3-319-46477-0.

OLGAC, A.; KARLIK, B. Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence And Expert Systems*, v. 1, p. 111–122, 02 2011.

PERICO, D. *USO DE HEURÍSTICAS OBTIDAS POR MEIO DE DEMONSTRAÇÕES PARA ACELERAÇÃO DO APRENDIZADO POR REFORÇO*. Tese (Doutorado), 11 2012.

RAMACHANDRAN, P.; ZOPH, B.; LE, Q. V. Searching for activation functions. *CoRR*, abs/1710.05941, 2017. Disponível em: <http://arxiv.org/abs/1710.05941>.

RANI, S. N.; RAO, S. Study and Analysis of Noise Effect on Big Data Analytics. *International Journal of Management, Technology And Engineering*, v. 8, n. 12, p. 5841–5850, 2019. Disponível em: <https://www.researchgate.net/publication/330508421>.

REDDY, Y.; PULABAIGARI, V.; B, E. Semi-supervised learning: a brief review. v. 7, p. 81, 02 2018.

ROSENBLATT, F. F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, v. 65 6, p. 386–408, 1958.

ROSHTKHARI, M. J.; LEVINE, M. D. An on-line, real-time learning method for detecting anomalies in videos using spatio-temporal compositions. *Comput. Vis. Image Underst.*, Elsevier Science Inc., USA, v. 117, n. 10, p. 1436–1452, out. 2013. ISSN 1077-3142. Disponível em: <https://doi.org/10.1016/j.cviu.2013.06.007>.

SHALEV-SHWARTZ, S.; BEN-DAVID, S. *Understanding machine learning : from theory to algorithms*. [s.n.], 2014. ISBN 9781107057135 1107057132. Disponível em: <http://www.worldcat.org/search?qt=worldcat_org_all&q=9781107057135>.

SPENCE, C.; PARRA, L.; SAJDA, P. Detection, synthesis and compression in mammographic image analysis with a hierarchical image probability model. *Proceedings of the Workshop on Mathematical Methods in Biomedical Image Analysis*, n. October, p. 3–10, 2001.

SRIVASTAVA, N. et al. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, v. 15, n. 56, p. 1929–1958, 2014. Disponível em: <http://jmlr.org/papers/v15/srivastava14a.html>.

SULTANI, W.; CHEN, C.; SHAH, M. Real-world anomaly detection in surveillance videos. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2018.

THEODORIDIS, S.; KOUTROUMBAS, K. *Pattern Recognition, Third Edition*. USA: Academic Press, Inc., 2006. ISBN 0123695317.

TIMMONS, N. G.; RICE, A. Approximating Activation Functions. 2020. Disponível em: <http://arxiv.org/abs/2001.06370>.

TURCHENKO, V.; LUCZAK, A. Creation of a deep convolutional auto-encoder in caffe. 12 2015.

VARGHESE, E.; MULERIKKAL, J.; MATHEW, A. Video anomaly detection in confined areas. *Procedia Comput. Sci.*, Elsevier Science Publishers B. V., NLD, v. 115, n. C, p. 448–459, nov. 2017. ISSN 1877-0509. Disponível em: <https://doi.org/10.1016/j.procs.2017.09.104>.

XU, D. et al. Learning deep representations of appearance and motion for anomalous event detection. In: . [S.l.: s.n.], 2015.

XU, D. et al. Learning deep representations of appearance and motion for anomalous event detection. In: . [S.l.: s.n.], 2015.

You, W. et al. An intelligent deep feature learning method with improved activation functions for machine fault diagnosis. *IEEE Access*, v. 8, p. 1975–1985, 2020.

Yun, K. et al. Motion interaction field for accident detection in traffic surveillance video. In: *2014 22nd International Conference on Pattern Recognition*. [S.l.: s.n.], 2014. p. 3062–3067. ISSN 1051-4651.