PREDIÇÃO DE PESO NA CRIAÇÃO DE FRANGO DE CORTE UTILIZANDO REDES NEURAIS ARTIFICIAIS

Eduardo Ferrari Ott, Andreza Sartori – Orientadora

Curso de Bacharel em Ciência da Computação
Departamento de Sistemas e Computação
Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brasil

efott@furb.br, asartori@furb.br

Resumo: *Este trabalho apresenta o desenvolvimento de um protótipo utilizando o modelo de rede neural Long-Short Term Memory para a predição de peso de frango baseado em variáveis ambientais. Foram utilizados dados de 17 lotes de frango, incluindo idade, peso, temperatura e umidade relativa. Na fase de pré-processamento dos dados foi utilizada a normalização de escala para controlar os valores dos pesos da rede durante o treinamento, e a reestruturação dos dados para adequá-la ao modelo LSTM. O modelo foi desenvolvido com o framework Keras, utilizando uma camada LSTM e uma camada intermediária, cada uma com 10 neurônios. O protótipo foi capaz de predizer o peso das aves com um Erro Quadrático Médio de 2,516 gramas. Este resultado mostra que o modelo desenvolvido poderia ser integrado a sistemas de gerenciamento das granjas para prever o peso das aves através de simulações com diferentes valores de temperatura e umidade, a fim de tornar a utilização dos sistemas de aquecimento mais eficiente.*

Palavras-chave*: Inteligência Artificial. Redes Neurais Artificiais. Frango de corte. Long Short Term Memory.*

1 INTRODUÇÃO

O Brasil está hoje entre os maiores produtores de carne de frango no mundo, junto com os Estados Unidos e a China. De acordo com a Associação Brasileira de Proteína Animal (2018), o Brasil produziu cerca de 13 milhões de toneladas de carne de frango em 2017, e exportou cerca de 4,45 milhões de toneladas, representando 36% da exportação mundial do produto.

Oliveira et al. (2012) apontam que a eficiência e dinamismo dessa atividade no país se deve ao constante ganho produtivo, alcançado através de avanços em pesquisa genética, aumento na automação da produção de aves e aperfeiçoamento de pessoal em relação ao manejo dos animais, além do sistema de produção integrado. Segundo Carvalho-Curi e Moura (2017 apud NAZARENO et al., 2009), estas melhorias estão relacionadas ao fato de que a avicultura de corte tem sempre buscado o progresso através de pesquisas em áreas como genética, instalações, manejo e conforto ambiental. Também mencionam que estes estudos objetivam compreender melhor os fatores que influenciam o desenvolvimento e o desempenho de frangos de corte para se obter a máxima produção de carne com o menor custo de produção considerando o bem-estar das aves.

Amaral et al. (2011) apontam que, dentre os diversos fatores que influenciam a produção de frangos de corte, os fatores ambientais, como a temperatura e a umidade relativa, são de grande importância durante o processo de criação, pois afetam diretamente a função vital mais importante das aves, a homeotermia. Aves mantidas em condições ambientais desfavoráveis, como excesso de calor, reduzem a sua ingestão de alimentos, o que prejudica o seu crescimento e a qualidade da carne, além de fazer com que a ave gaste mais energia para promover a perda desse calor (DOZIER et al. 2006; LU et al. 2007). Pereira e Nääs (2005) apontam que nos diversos estudos sobre a termoneutralidade de aves, observa-se uma grande variação entre os valores limítrofes atualmente indicados para a criação. Essa variação ocorre devido às diferenças genéticas, climáticas e adaptativas das aves em diferentes cenários.

Levando em conta estes aspectos, a necessidade do controle e entendimento do ambiente interno nas instalações avícolas é a principal razão do uso de diferentes métodos matemáticos e computacionais, tais como a lógica fuzzy e redes neurais artificiais (RNAs) (CARVALHO-CURI; MOURA, 2017). Dentro desses métodos, as RNAs têm sido utilizadas amplamente para a identificação de padrões para a predição de dados em séries temporais (FERRAZ et al., 2014). Entretanto, as RNAs convencionais, quando utilizadas em cenários com intervalos de tempo extensos provocam o desaparecimento de gradiente, limitando a capacidade de treinamento do modelo de RNA desenvolvido. Como solução para este problema, Hochreiter e Schmidhuber sugerem a utilização do modelo Long Short Term Memory (LSTM), que cria 'pontes' entre as séries temporais para impedir a perda de gradiente e ineficácia da rede (HOCHREITER; SCHMIDHUBER, 1997).

Considerando o exposto, este trabalho tem como objetivo desenvolver e treinar um protótipo para a predição de peso de frango de corte utilizando o modelo de rede neural recorrente LSTM. O trabalho tem como base para o treinamento da rede fatores ambientais na criação das aves, como a temperatura e a umidade relativa, a fim de compreender a influência desses fatores sobre o crescimento dos animais. Desta forma, este trabalho visa contribuir

com o processo de criação avícola, validando a aplicabilidade do modelo LSTM na predição de peso das aves com a finalidade de auxiliar o produtor no que se refere a decisões estratégicas, que dependam do peso das aves. Com isso, este protótipo tem potencial para ser utilizado como ferramenta para avaliar a eficiência do controle de ambiência das granjas, assim como, pode ser integrado a sistemas de controle de microclima, para otimizar o controle da ambiência das aves.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem como objetivo explorar os assuntos que fundamentaram o estudo realizado. A seção 2.1 aborda a ambiência na criação de frango de corte, enquanto a seção 2.2 descreve os principais conceitos de redes neurais.

2.1 AMBIENCIA NA CRIAÇÃO DE FRANGO DE CORTE

Atualmente, o maior custo na produção de aves de corte é a ração consumida por elas durante a sua criação, representando cerca de 65% do custo total, enquanto o peso médio das aves é o maior responsável pela diluição desse custo (cerca de 35% do custo total). Esses dois indicadores podem ser relacionados através de uma medida de desempenho, a Conversão Alimentar (CA) que é definida pelo consumo total de ração pelas aves, dividido pelo seu peso médio. Muitas companhias utilizam a CA como a principal medida de desempenho durante a criação das aves buscando compreender os fatores básicos que afetam esses dois índices, a fim de aumentar o rendimento das aves (TORRETTA, 2017).

A evolução da avicultura resultou em um frango de corte precoce, com grande eficiência para converter diferentes alimentos em proteína. Apesar disso, uma série de problemas relacionados ao metabolismo das aves e ao manejo nas granjas tem surgido, destacando-se entre eles o estresse calórico. A suscetibilidade das aves ao estresse calórico aumenta à medida que a umidade relativa e a temperatura ambiente ultrapassam a zona de conforto térmico, dificultando assim a dissipação de calor e consequentemente incrementando a temperatura corporal da ave, causando um efeito negativo sobre o ganho de peso (BORGES et al. 2003).

De acordo com Amaral et al. (2011), “o ambiente de produção exerce papel fundamental na avicultura moderna, que busca alcançar alta produtividade, em espaço físico e tempo relativamente reduzidos”. Os autores frisam ainda que, dentre todos os fatores que influenciam o crescimento das aves, a temperatura e umidade do aviário estão entre os que mais afetam o seu desempenho. Isso porque as aves atingem o seu pico de produtividade (maior desempenho de CA) quando mantidos em ambiente termoneutro, ou seja, quando a energia proveniente da alimentação não é desviada para compensar desvios térmicos em relação ao intervalo de termoneutralidade para eliminar ou manter o seu calor (CARVALHO-CURI; MOURA, 2017).

Para aumentar o desempenho das aves, de acordo com o estudo de Carvalho (2017), vários autores procuram estabelecer temperaturas ideais para o ambiente de criação. Essas temperaturas são classificadas em intervalos definidos por limites de temperatura inferiores e superiores, considerando as diferentes idades das aves (Quadro 1). Entretanto, esses intervalos variam de acordo com o tipo das aves, posição geográfica entre outras variáveis que afetem a sensação térmica, isto é o conforto térmico, das aves.

Quadro 1 - Limites inferiores e superiores das temperaturas do ar ideais de criação de frangos de corte em diferentes semanas de vida.

Semana	Limites de temperatura do ar (°C)
Primeira	34 – 32
Segunda	32 – 28
Terceira	28 – 26

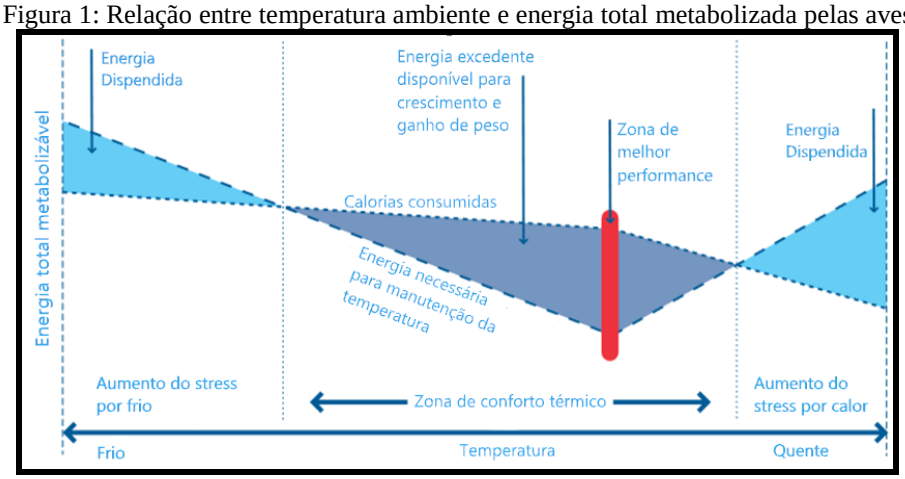
Fonte: Carvalho (2017).

O estresse calórico tem um maior impacto nos primeiros dias de vida das aves, pois, por não possuírem o sistema termorregulatório totalmente desenvolvido, os pintinhos precisam ser criados em ambientes com a temperatura controlada nas primeiras duas semanas de vida, ou até quatro semanas em regiões com condições ambientais mais rigorosas (MENEGALI et al., 2013). De fato, condições térmicas às quais as aves são submetidas nestas primeiras semanas de vida são importantes para o desenvolvimento futuro do animal, sendo necessário monitorar as instalações para o controle do ambiente térmico uma vez que fatores térmicos podem comprometer o seu desempenho (OLIVEIRA et al., 2012).

A figura 1 mostra a relação entre a temperatura e eficiência do metabolismo das aves ao converter alimento em proteína (ganho de peso). Pode-se observar que, à medida que a temperatura aumenta ou diminui em relação a zona de conforto ideal para as aves (zona de melhor performance), maior se torna a quantidade de energia necessária para a manutenção da temperatura corporal das aves. Também pode-se verificar que após a temperatura ultrapassar a zona de

conforto das aves, a energia dissipada para essa manutenção se torna maior do que a energia proveniente da alimentação do animal resultando na sua perda de peso.

Figura 1: Relação entre temperatura ambiente e energia total metabolizada pelas aves



Fonte: Baseado em Aviagen (2010).

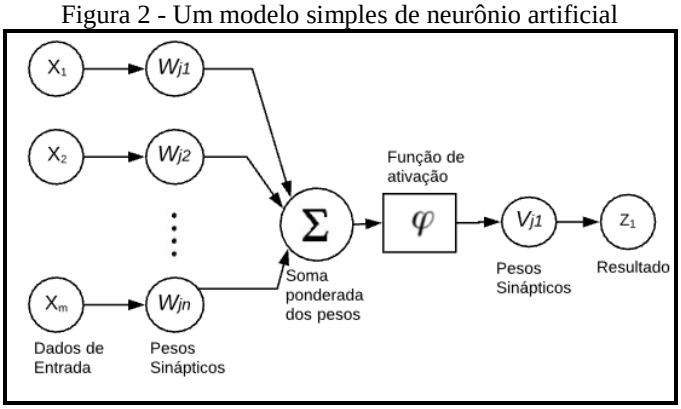
Os sistemas automatizados de climatização utilizados para o controle desses ambientes são capazes de controlar diversas variáveis em tempo real a fim de garantir que a temperatura se mantenha próxima a níveis ideais para os animais. Nesse contexto, o desenvolvimento de algoritmos e sistemas para controle de ambiência baseados em variáveis meteorológicas, possíveis de serem mensuradas automaticamente e de forma não invasiva, podem propiciar melhoria no desempenho produtivo dos animais (CARVALHO, 2017).

2.2 REDES NEURAIS ARTIFICIAIS

Haykin (2007) define uma Rede Neural Artificial (RNA) como um processador paralelamente distribuído constituído de unidades de processamento simples, que têm a propensão natural para armazenar conhecimento experimental e torná-lo disponível para o uso. De fato, o objetivo das RNAs é modelar funções matemáticas complexas, de forma que para cada entrada se obtenha um resultado esperado (GRAVES, 2008).

A estrutura das redes neurais consiste em uma determinada quantidade de unidades chamadas neurônios, que são conectados a outros neurônios por links diretos de comunicação. Cada neurônio possui um peso associado e são estruturados em diferentes camadas conectadas entre si. Cada neurônio possui um estado interno, também chamado de sua 'ativação', que é uma função das entradas que recebeu. Um neurônio normalmente envia um sinal para vários outros neurônios bem como o recebe de diversos outros (FAUSETT, 1994). A figura 2 mostra a estrutura básica de um neurônio, no qual a função de ativação recebe como entrada a soma ponderada entre as entradas (X_n) e os pesos (W_n). Esta função define se o sinal recebido irá ou não ativar o neurônio, propagando o sinal para camadas posteriores.

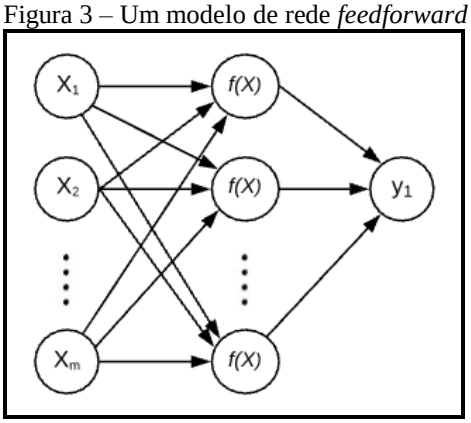
Figura 2 - Um modelo simples de neurônio artificial



Fonte: Baseado em Fausett (1994).

Uma rede neural pode ter três tipos de camadas, uma camada de entrada, onde são informados os parâmetros para a computação da rede, uma ou mais camadas intermediárias, que fazem a computação dos valores de entrada com os pesos das sinapses, e uma camada de saída, que computa o resultado esperado pela rede. A topologia mais simples de RNA são as redes *feedforward*, na qual a saída de cada neurônio serve como entrada para os neurônios da próxima camada. A figura 3 ilustra um modelo simples de rede neural *feedforward*.

Figura 3 – Um modelo de rede *feedforward*



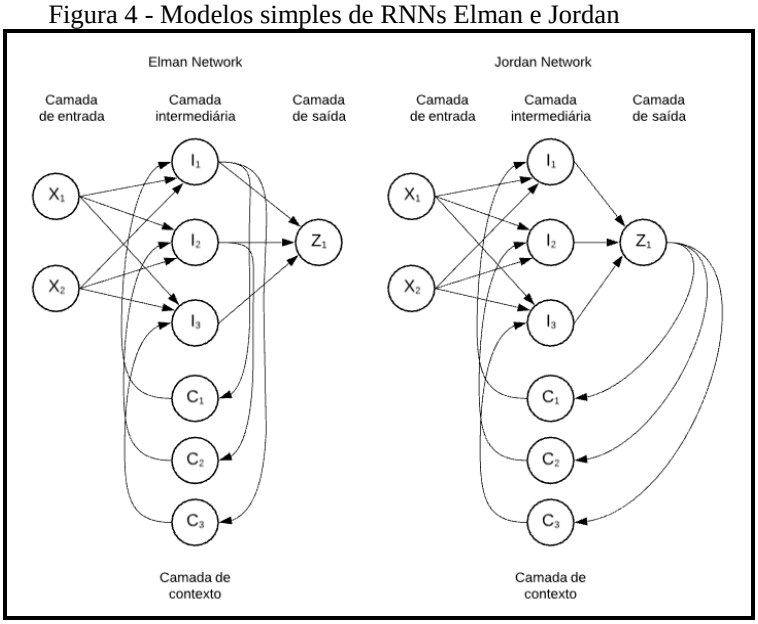
Fonte: Baseado em Fausett (1994).

Para seu uso, as RNAs precisam passar por uma etapa de treinamento. No caso de redes neurais treinadas de forma supervisionada, este treinamento é realizado passando à rede exemplos de entradas e seus respectivos resultados esperados. O treinamento de uma RNA é realizado em duas etapas. Na primeira etapa, chamada *feedforward*, cada unidade de entrada recebe um sinal e propaga este sinal aos neurônios da primeira camada intermediária, cada qual computa a sua função de ativação e propaga um novo sinal aos neurônios da camada seguinte, até que o valor final seja computado pela unidade de saída. Já na segunda etapa, chamada *backpropagation*, o valor resultante do exemplo testado é comparado ao resultado esperado, gerando uma função de custo, que é utilizada para propagar o erro computado às camadas anteriores atualizando os pesos de cada conexão entre as camadas de entrada, intermediárias e de saída. (FAUSETT, 1994).

Após o seu treinamento. É possível avaliar a performance da RNA. Entre as métricas mais comuns para mensurar a performance de uma rede, está o Erro Quadrático Médio (Root Mean Squared Error – RMSE). Para definir o erro quadrático médio de uma rede, são utilizados os resultados reais de cada observação e os resultados preditos pela rede. O RMSE consiste na raiz da diferença entre soma dos valores reais e dos valores preditos, dividida pela quantidade de predições. O valor computado do RMSE sempre é apresentado na mesma unidade de medida dos valores preditos. (TWOMEY; SMITH, 1995)

Para a predição de séries temporais, são utilizadas redes neurais recorrentes (Recurrent Neural Networks – RNN), que tem por base uma rede neural *feedforward* com algumas modificações, como a realimentação (GOMES, 2005). A realimentação se trata da reutilização dos valores computados pela rede, criando loops e fazendo com que a rede seja executada recursivamente para cada passo de tempo. (MEDSKER, JAIN, 2001). Existem diversas topologias de RNNs que variam de acordo com cada necessidade. Na figura 4 são apresentados dois exemplos de RNNs. No modelo à esquerda, de Elman, o estado dos neurônios na camada intermediária (I_n) é guardado na camada de contexto (C_n) e reutilizado nas camadas intermediárias. No segundo modelo, de Jordan, o resultado computado pela rede é armazenado na camada de contexto (C_n), e este é reutilizado pelas camadas intermediárias (I_n).

Figura 4 - Modelos simples de RNNs Elman e Jordan



Fonte: Elaborado pelo autor baseado em Medsker e Jain (2001).

Na fase de treinamento, as redes neurais recorrentes utilizam o algoritmo de retropropagação através do tempo (Back-Propagation Through Time – BPTT) que é uma extensão do algoritmo de retropropagação padrão (GOMES, 2005). Este tipo de rede, no entanto, possui limitações ao trabalhar com grandes sequencias de dados nas quais ocorre o problema do desaparecimento de gradiente (*vanishing-gradient problem*). Este problema ocorre quando as primeiras camadas da rede perdem a sua capacidade de treinamento devido aos valores dos ajustes dos pesos se tornarem extremamente pequenos, não representando mudanças significativas da capacidade de predição do modelo (HOCHREITER, SCHMIDHUBER, 1997)

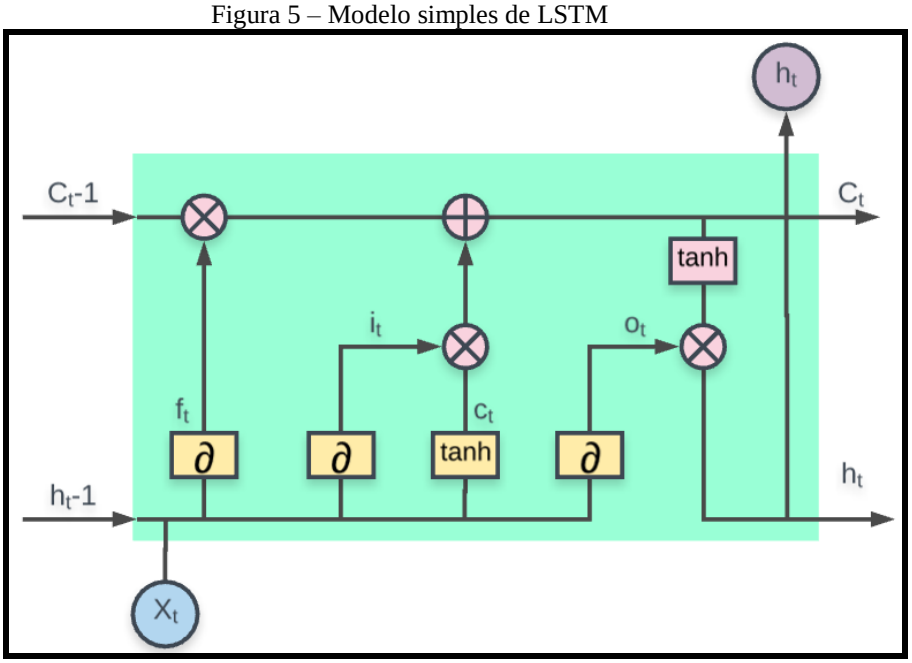
2.2.1 Rede Neural Recorrente LSTM

Uma rede Long Short-Term Memory (LSTM) é um modelo recorrente e profundo de redes neurais que utiliza unidades especiais chamadas "portões" para resolver o problema de desaparecimento de gradiente ao trabalhar com grandes sequencias. Este modelo busca manter um fluxo constante de erro através destas unidades, utilizando-os para ignorar partes da sequência de dados que o modelo não considere mais necessárias ou que possam comprometer o seu treinamento, mantendo apenas a "memória" importante para o treinamento (HOCHREITER, SCHMIDHUBER, 1997).

A estrutura básica de uma rede LSTM é similar a uma RNN por se estruturar em uma cadeia durante o seu treinamento. Entretanto, uma RNN normalmente necessita de apenas uma camada intermediária adicional para armazenar memória a cada passo de tempo, limitando a sua capacidade de "memorização" enquanto a LSTM possui quatro camadas de rede, cada uma com diferentes funções (COLAH, 2015).

Colah (2015) aponta que a chave para o funcionamento de uma LSTM é o estado da célula, que é alterado de acordo com a passagem dos pesos através da rede. Esse estado transporta as possíveis predições computadas para cada observação computada, ou seja, para cada peso predito, assim como a "memória" da rede, ou seja, o peso de cada possível predição. A LSTM pode adicionar ou remover informações a este estado, o que é regulado pelas estruturas dos portões. Os portões são uma maneira de controlar a informação que é transmitida entre as redes por esse estado, e são compostos cada um por uma rede neural sigmoide (representados na figura 5 por ∂) e uma operação de multiplicação de vetores.

Figura 5 – Modelo simples de LSTM



Fonte: Baseado em Colah (2015).

A primeira etapa da LSTM é decidir que dados (predições) devem ser removidos do estado da célula. Essa decisão é feita pela primeira rede sigmóide chamada de *forget gate layer*, esta camada recebe as predições do passo anterior (h_t-1) e os novos dados de entrada do passo atual, e os computa em um valor entre 0 e 1 para cada dado no estado da célula (C_t-1). Um valor igual a 1 mantém os dados de entrada no estado, enquanto o valor 0 remove os dados do estado, "ignorando" predições que a rede considere irrelevantes para o passo atual.

Na segunda etapa, a rede decide que informações serão mantidas no estado da célula. Isso ocorre em três partes, primeiro a segunda camada sigmoide, chamada *input gate layer* decide que informações serão atualizadas. Em seguida, a camada `tanh` cria um vetor de novos candidatos (C_t), que poderiam ser adicionados ao estado. Por fim, o produto das duas camadas é somado ao estado da célula, atualizando as predições da rede e a sua "memória", resultando no estado que será enviado para o próximo passo.

Por fim, a rede decide qual será o seu output com base no estado da célula. Para isso, há uma última camada sigmoide que computa quais serão as possíveis saídas (predições), o estado da célula é então transformado com a função `tanh` (para manter todos os valores entre 1 e -1) e o multiplica pela saída da camada sigmoide, de maneira que o output seja apenas as possíveis predições computadas pela rede.

2.3 TRABALHOS CORRELATOS

Neste capítulo serão apresentados três trabalhos correlatos à proposta. O primeiro trabalho, desenvolvido por Carvalho (2017), apresentado no quadro 2 é um estudo sobre a aplicação de redes neurais na busca por predições de temperatura superficial durante a criação de frangos de corte. O segundo, de Faridi et al. (2013), apresentado no quadro 3, aborda o desenvolvimento e resultados de modelos de redes neurais para a predição de peso de ovos de matrizes, isto é, de galinhas poedeiras. Por fim, o trabalho de Ferraz et al. (2014), apresentado no quadro 4, faz a predição da massa corporal de pintinhos por meio de redes neurais e redes-fuzzy neurais.

Quadro 2 - Trabalho Correlato 1

Referência	Carvalho (2017)
Objetivos	Aplicação de redes neurais na predição da temperatura superficial de pintinhos entre 1 e 14 dias de vida.
Principais funcionalidades	Predição da temperatura superficial de pintinhos entre 1 e 14 dias de vida utilizando modelos de rede neural.
Ferramentas de desenvolvimento	Rede neural Multicamadas.
Resultados e conclusões	O modelo treinado utilizando 10 neurônios na camada escondida apresentou os melhores resultados, tendo um Erro Quadrático Médio de 0,4741C° na etapa de treinamento e 0,7452 C° na etapa de testes, mostrando que a rede pode predizer com alta confiabilidade a temperatura superficial das aves. O autor sugere que com base nos dados resultantes do melhor modelo seria possível desenvolver softwares integrados às controladoras de temperatura dos aviários a fim de melhorar o controle do ambiente térmico durante o seu ciclo produtivo.

Fonte: elaborado pelo autor.

O trabalho de Carvalho (2017) relaciona-se ao presente trabalho por utilizar modelos de redes neurais bem como utilizar dados ambientais obtidos durante a criação de aves de corte para predição de características das aves. Carvalho (2017) buscou predizer a temperatura corporal das aves, enquanto neste trabalho propõe-se predizer o seu peso.

Quadro 3 - Trabalho Correlato 2

Referência	Faridi et al. (2013)
Objetivos	Modelo de rede neural para predição do ganho de peso diário de ovos de matrizes de frango
Principais funcionalidades	Predição do peso de ovos produzidos por matrizes de frango no início da etapa de postura, entre a 25ª e 28 ª semanas de vida, em função dos níveis nutricionais das aves.
Ferramentas de desenvolvimento	Rede Neural Multicamadas.
Resultados e conclusões	O modelo treinado utilizando 6 neurônios na camada escondida apresentou os melhores resultados, tendo um Erro Quadrático Médio de 0,39Kg na etapa de treinamento e 0,025Kg na etapa de testes, mostrando que a rede pôde estimar efetivamente o peso dos ovos. O autor sugere que com base nos dados resultantes do melhor modelo, seria possível simular diferentes cenários a fim de identificar a melhor combinação de nutrientes para maximizar o peso dos ovos produzidos nas semanas iniciais de postura das aves.

Fonte: elaborado pelo autor.

O trabalho de Faridi et al. (2013) também se relaciona ao presente trabalho por utilizar redes neurais. O modelo também busca a predição de peso, porém, de ovos, enquanto o trabalho atual busca predizer o peso das aves.

Quadro 4 - Trabalho Correlato 3

Referência	Ferraz et al. 2014
Objetivos	Predição da massa corporal de pintinhos por meio de redes neurais e redes neuro-fuzzy.
Principais funcionalidades	Predição da massa corporal de pintinhos entre 2 a 21 dias de idade, sujeitos a diferentes durações e intensidades de estresse térmico
Ferramentas de desenvolvimento	Rede Neural Multicamadas e Rede Neuro-Fuzzy
Resultados e conclusões	O modelo treinado utilizando 100 neurônios na camada escondida apresentou os melhores resultados dentre os modelos de rede neural convencionais, tendo um Erro Quadrático Médio de 01663Kg na etapa de testes. Já dentre os modelos de rede Fuzzy-Neural, o modelo treinado utilizando 27 regras apresentou os melhores resultados, tendo um Erro Quadrático Médio de

	0,0265Kg na etapa de testes. O autor conclui que com base nos dados resultantes do melhor modelo seria possível agrega-lo a um software integrado à uma controladora a fim de melhorar o controle do ambiente térmico durante a criação das aves.

Fonte: elaborado pelo autor.

O trabalho de Ferraz et al. (2014) também se relaciona ao trabalho atual pela utilização de redes neurais para predição de peso das aves, no qual utiliza modelos de RNAs multicamadas e Neuro-fuzzy. Estas redes por padrão realizam predições com base em pares de valores (os parâmetros de entrada e o resultado esperado), sem levar em conta os passos anteriores do treinamento da rede. O presente trabalho propõe a utilização do modelo LSTM, que armazena o "histórico" de predições passadas da rede durante o seu treinamento, oferecendo mais precisão quando utilizado para predição de séries temporais.

3 ESPECIFICAÇÃO E DESENVOLVIMENTO DO PROTÓTIPO

Nesta seção serão apresentados os aspectos mais relevantes relacionados à especificação e desenvolvimento do protótipo para a predição de peso de frango de corte. São abordados os requisitos do protótipo, seguidos de sua especificação e desenvolvimento do modelo.

3.1 REQUISITOS

O protótipo desenvolvido atenderá os Requisitos Funcionais (RF) e Requisitos Não Funcionais (RNF apresentados no Quadro 5 e Quadro 6, respectivamente. Os requisitos são mapeados com os casos de uso ilustrados na Figura 6.

Quadro 5 – Requisitos funcionais

Requisitos Funcionais (RF)
RF01: permitir que o usuário carregue uma base de dados ao sistema
RF03: realizar o treinamento da rede neural utilizando os dados carregados pelo usuário
RF04: predizer o peso das aves com base nos dados carregados pelo usuário

Fonte: elaborado pelo autor.

Quadro 6 – Requisitos não funcionais

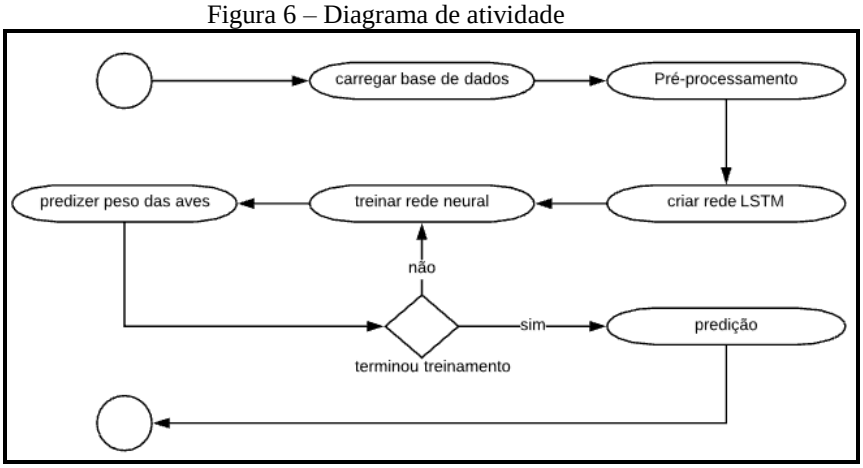
Requisitos Funcionais (RF)
RNF01: ser desenvolvido utilizando a linguagem python
RNF02: utilizar o framework Keras para desenvolvimento e treinamento do modelo

Fonte: elaborado pelo autor.

3.2 DESENVOLVIMENTO DO PROTÓTIPO

Esta seção descreve o diagrama de atividades do protótipo, apresentado na figura 7, e detalha as fases de pré-processamento dos dados e a implementação do modelo de rede neural LSTM. A primeira etapa do módulo é o carregamento da base de dados para o protótipo. A etapa seguinte realiza o pré-processamento dos dados carregados, incluindo a sua normalização e reestruturação para um formato de aprendizado supervisionado, necessário para carregar a base de dados para a rede neural LSTM. Então, a rede neural LSTM é criada e parametrizada. Com a rede criada, é realizado o seu treinamento utilizando os dados pré-processados. Durante a etapa de treinamento são realizadas as predições e ajustes dos pesos de acordo com os parâmetros estabelecidos. Ao final do treinamento, o protótipo retorna o modelo contendo a rede para predição de pesos com base nos dados carregados pelo usuário.

Figura 6 – Diagrama de atividade



Fonte: elaborado pelo autor.

3.2.1 Pré-processamento da Base de Dados

Para realizar o treinamento das redes implementadas, foram utilizados dados obtidos de 17 lotes de aves contendo entre dez mil e cinquenta mil aves cada lote. A base de dados para o treino possui informações características do ciclo de criação dos lotes para cada dia de vida entre o primeiro e vigésimo sétimo dia de idade, totalizando 459 observações. Para cada idade, obteve-se o peso médio das aves, a temperatura e umidade relativa médias do galpão onde as aves se encontravam. A tabela 1 mostra um exemplo dos dados utilizados para o treinamento, nela é possível observar as primeiras 10 linhas da base de dados, as colunas são respectivamente o número do lote, a idade das aves, o seu peso em gramas, a temperatura em Cº e a umidade relativa.

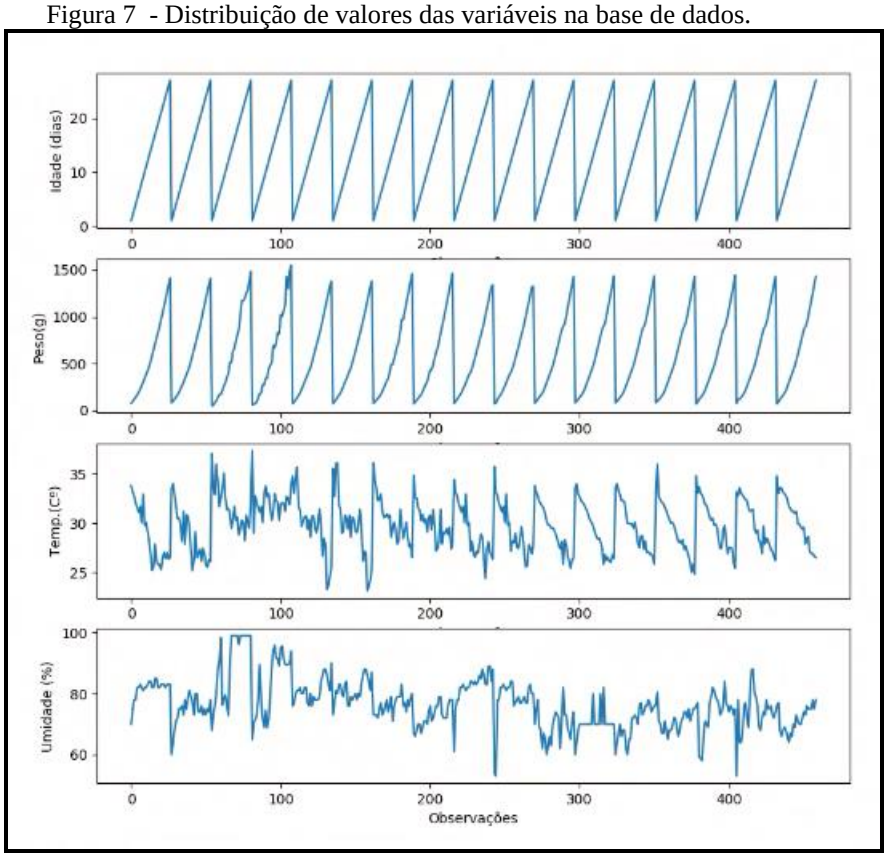
Tabela 1 – Base de Dados parcial utilizado para treinamento dos modelos

LOTE	IDADE (dias)	PESO (gramas)	TEMPERATURA (Cº)	UMIDADE RELATIVA(UR%)
1	1	77.5	33.81	70
1	2	95	33.16	75
1	3	117.5	32.79	78
1	4	140.5	32.05	78
1	5	166	31.6	82
1	6	191	31.14	82
1	7	228.5	31.69	83
1	8	260.5	30.06	82
1	9	301	32.97	81
1	10	341.5	29.79	82

Fonte: elaborado pelo autor.

A figura 8 mostra a distribuição de cada variável através de toda a base de dados, isto é, o valor de cada variável através das 459 observações, ou exemplos. Cada quadro apresenta respectivamente as idades dos lotes, entre 1 e 27 dias, o peso, a temperatura e a umidade relativa em cada dia. Observando os dados é possível perceber que não há ruído nos dados utilizados, isto é, valores muito altos ou muito baixos em relação ao valor médio de cada variável.

Figura 7 - Distribuição de valores das variáveis na base de dados.



Fonte: elaborado pelo autor.

Para carregar e adequar a base de dados de treino à rede implementada, foram utilizadas funções das bibliotecas Pandas para carregamento e manipulação de tabelas, Sci-kit Learn para o pré-processamento dos dados e avaliação da performance do modelo além da biblioteca Numpy, para a manipulação de matrizes. O quadro 7 apresenta o código implementado para o carregamento, pré-processamento e preparação dos dados para treinamento do modelo. Para carregar base de dados, foi utilizada a função `read_csv` da biblioteca Pandas. A primeira etapa na preparação do dataset foi a conversão dos dados para o formato `float u`tilizando a função `asType`, para garantir que todos os dados estejam no formato necessário para os passos seguintes. Após isso, os dados foram normalizados através da função `MinMaxScaller`. Esta função converte os valores de cada coluna da base de dados para um valor entre 0 e 1, sendo 0 o menor valor da coluna, e 1, o maior. Essa normalização possibilita um aprendizado mais eficiente da rede, permitindo ao modelo criar uma relação mais concisa entre as variáveis envolvidas, assim como evitar que atributos com variáveis com valores muito distintos impactem na capacidade de predição do modelo durante o treinamento (LECUN, 1998).

A base de dados foi então reestruturada para um modelo de aprendizado supervisionado. Para isso, foi utilizada a função `series_to_supervised`, que reestrutura a base de dados adicionando uma coluna na qual é inserido o peso das aves do dia seguinte de cada observação, ou seja, o peso que se deseja predizer. Dessa forma, ao passar os dados para o treinamento da rede, esta coluna adicional será definida como a saída, ou predição esperada, de cada observação.

A base de dados foi então dividida em duas partes, a primeira para treinamento e a segunda para testes. O percentual para divisão dos dados foi definido através de testes com valores percentuais entre 55% e 80%, sendo que o modelo que utilizou 65% dos dados para treino obteve os melhores resultados. Desta forma, foi feita a divisão para treino de 65% dos dados disponíveis na base de dados, totalizando 297 observações, e para testes, foram utilizadas as 162 observações restantes, representando 35% dos dados. Esta divisão foi feita levando em conta cada lote individual, ou seja, separando a base de dados de 27 em 27 linhas (ciclo de vida das aves). Esta divisão em grupos (*batches*) foi parametrizada posteriormente ao carregar os dados para a rede.

Por fim, os dados foram separados em duas matrizes e dois vetores. As duas matrizes receberam os valores de entrada para treino e testes, enquanto os dois vetores receberam as saídas esperadas para cada exemplo de treino e testes. Além disso, a propriedade `shape` das matrizes foi alterada para identificar a quantidade de observações, passos e dados de entrada de cada uma, essas informações são passadas posteriormente à rede para o treinamento.

Quadro 7- Carregamento, pré-processamento e preparação dos dados

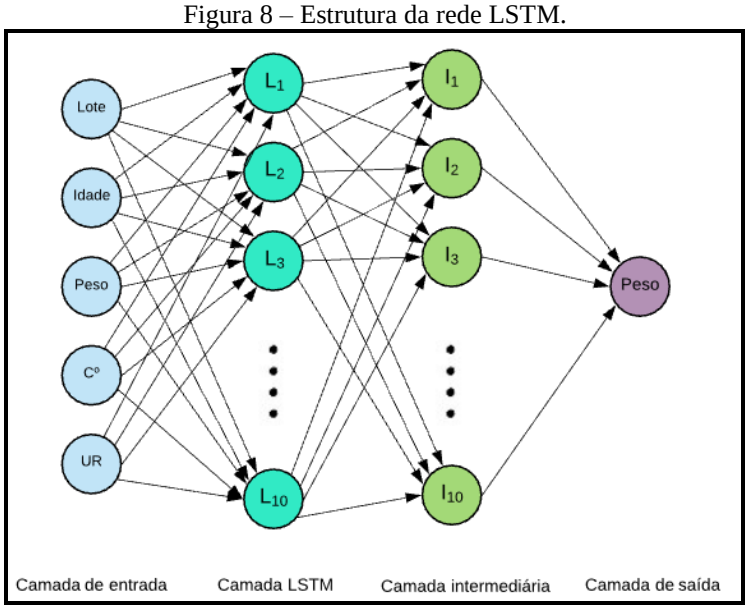
```
# Preparação dos dados
# Carrega base de dados
dataset = read_csv('LSTMRNN-V2-2_Bkp.csv', header=0, index_col=0)
values = dataset.values

# Pesos salvos separadamente para posterior desnormalização
values2 = dataset.values

# Converte todos os dados para float
values = values.astype('float32')
# Normaliza os dados entre 0 e 1
scaler = MinMaxScaler(feature_range=(0, 1))
scaled = scaler.fit_transform(values)

# Normaliza os pesos separadamente para desnormalização
scaler2 = MinMaxScaler(feature_range=(0, 1))
values_2 = values[:, 2].reshape(-1, 1)
scaled2 = scaler2.fit_transform(values_2)
# Estrutura os dados para aprendizado supervisionado
reframed = series_to_supervised(scaled, 1, 1)
values = reframed.values

# Remove colunas que não serão utilizadas
reframed.drop(reframed.columns[[5, 6, 8, 9]], axis=1, inplace=True)
values = reframed.values

# Divide os dados entre treino e teste
n_train_hours = int(len(dataset) * 0.65)
train = values[0:n_train_hours, :]
test = values[n_train_hours:len(dataset), :]

# Divisão entre entradas e resultados
train_X, train_y = train[:, :-1], train[:, -1]
test_X, test_y = test[:, :-1], test[:, -1]
# Reestruturação dos metadados das tabelas para identificar
# a quantidade de observações, passos e dados de entrada
train_X = train_X.reshape((train_X.shape[0], 1, train_X.shape[1]))
test_X = test_X.reshape((test_X.shape[0], 1, test_X.shape[1]))
```

Fonte: elaborado pelo autor.

3.2.1 Desenvolvimento do modelo de predição de peso

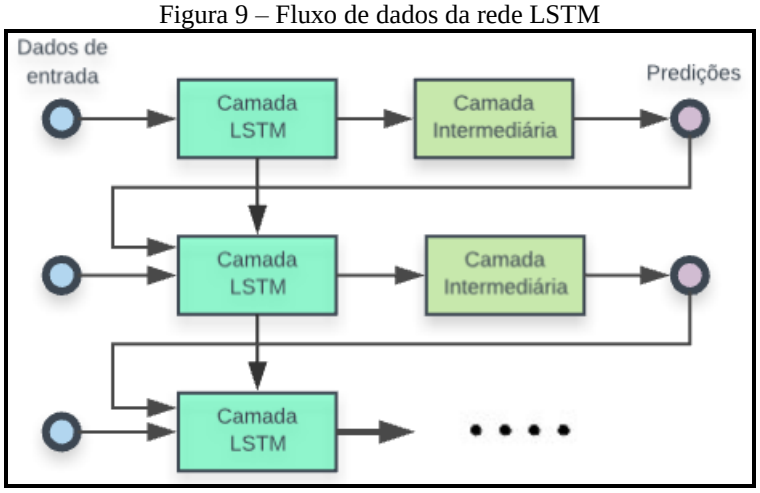
Na figura 9 é apresentada a estrutura da rede LSTM enquanto a figura 10 apresenta o fluxo de dados entre as camadas e iterações da rede. Os dados de entrada passam pela camada LSTM e são propagados à camada intermediária, que computa a predição da observação atual. Esta predição é adicionada à entrada da próxima execução, ou seja, para o próximo peso que será predito. Além disso, o estado interno da camada LSTM é transportado de uma execução para a próxima, mantendo a sua ‘memória’.

Figura 8 – Estrutura da rede LSTM.



Fonte: elaborado pelo autor.

Figura 9 – Fluxo de dados da rede LSTM



Fonte: elaborado pelo autor.

O quadro 8 ilustra a implementação da rede com duas camadas, sendo uma a camada LSTM e a segunda uma camada intermediária (neste caso, não sendo LSTM), além da camada de saída. Para a criação do protótipo, foi utilizado o modelo `Sequential` da biblioteca Keras. Este modelo se trata de uma pilha linear de camadas, ou seja, a computação dos dados ocorre de maneira sequencial, onde a entrada de cada camada será a saída da camada anterior. A primeira camada instanciada foi a camada LSTM, a qual foi criada com 10 neurônios. A quantidade de neurônios foi definida com base em testes utilizando entre 5 e 100 unidades, apresentados na seção de resultados.

No modelo `Sequential`, a primeira camada deve receber os parâmetros que definirão a forma da entrada de dados. Dessa forma, a camada LSTM recebeu, como parâmetros de `input_shape`, os *timesteps* `(train_X.shape[1])` que informam à rede a quantidade de observações anteriores que se devem considerar como entrada, ou seja, apenas os dados do dia anterior, e as características `(train_X.shape[2])`, que informam à rede a quantidade de variáveis de entrada (lote, idade, peso, temperatura e umidade do dia anterior), ou seja, 5 características. A função de ativação definida para esta camada foi a função `sigmoid`, que apresentou melhores resultados em relação às demais funções testadas (`tanh` e `relu`). A função `sigmoid` garante a normalização dos pesos durante o treinamento da rede, mantendo-os com valores constantes entre 0 e 1, para evitar que os valores desses pesos se ajustem a valores muito altos que possam resultar em erros durante o treinamento da rede.

Quadro 8 – Criação da rede neural LSTM

```
#Criação do modelo
model = Sequential()
# Camada LSTM
model.add(LSTM(10, activation='sigmoid', input_shape=(train_X.shape[1],
train_X.shape[2])))
# Camada intermediária
model.add(Dense(10, activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(1))
model.compile(loss='mae', optimizer='adam')
```

Fonte: elaborado pelo autor.

Posterior à camada LSTM, foi adicionada uma segunda camada intermediária com 10 neurônios. Esta camada recebe as predições computadas pela camada LSTM para computar a predição final de cada observação. Para esta camada, foi utilizada a função de ativação `tanh` que, similarmente à função `sigmoid` garante a normalização dos pesos durante o treinamento da rede. A função `tanh`, entretanto, mantém os valores dos pesos entre -1 e 1 para evitar que estes valores se tornem muito altos.

Adicionalmente, foi utilizada a técnica `dropout` na camada oculta para otimizar o treinamento da rede. A técnica `dropout` faz com que, durante o treinamento, alguns neurônios sejam 'desligados' a cada iteração, juntamente com as suas conexões, com o objetivo de prevenir que o modelo se ajuste ao conjunto de dados de forma que seja ineficaz ao prever novos resultados, isto é, visa evitar o *overfitting*. Por fim, foi instanciada uma camada de saída com um único neurônio, visto que o resultado deve ser um valor de peso individual para cada observação.

Antes do treinamento, foram definidas as configurações para o modelo através do método `compile`, sendo a função de perda, definida como Erro Absoluto Médio (MAE) e o otimizador `adam`. A função de perda tem como objetivo calcular a variação de erro durante o treinamento enquanto o otimizador `adam` realiza o ajuste dos pesos com base nessas variações.

O quadro 9 ilustra a configuração da rede, bem como para apresentação dos dados da função de perda a cada iteração. No treinamento do modelo foram utilizados dois parâmetros principais, `epochs` (epocas) e `batch_size`, além das matrizes com os parâmetros de entrada e os vetores de resultados esperados. O parâmetro de épocas define a quantidade de iterações que a rede executará sobre a base de dados, na qual foram aplicadas 50 épocas.

Quadro 9 – Treinamento da rede e plotagem da função de perda para cada iteração

```
# Treinamento da rede
history = model.fit(train_X, train_y, epochs=50, batch_size=27,
                    validation_data=(test_X, test_y))

# Apresentar dados de função de perda
pyplot.ylim(0, 0.35)
pyplot.plot(history.history['loss'], label='train')
pyplot.plot(history.history['val_loss'], label='test')
pyplot.ylabel('taxa de perda')
pyplot.xlabel('épocas')
pyplot.legend()
pyplot.show()
```

Fonte: elaborado pelo autor.

O parâmetro `batch_size` representa o intervalo de idade das aves, ou seja, o intervalo entre as observações que devem ser agrupadas durante o treinamento. Desta forma, o parâmetro foi definido com o valor 27, que é o intervalo de idade das aves na base de dados utilizada.

O quadro 10 ilustra a execução das predições, a desnormalização dos dados e o cálculo do RMSE. Os dados de treino e teste foram utilizados para realizar as predições com o modelo treinado. Para isso foi utilizado o método `predict`, passando como parâmetro as matrizes com os valores de entrada de treino e de testes. As predições geradas pelo modelo foram então desnormalizadas, isto é, os valores foram convertidos das escalas de 0 a 1 para suas escalas originais, de forma que fosse possível fazer a compararação dos resultados. Com os resultados reais e as predições da rede neural, foi possível avaliar a performance da rede através do Erro Quadrático Médio (Root Mean Squared Error – RMSE).

Quadro 10 – Treinamento, desnormalização e cálculo do RMSE

```
# Realiza predições
trainPredict = model.predict(train_X)
testPredict = model.predict(test_X)

# Desnormaliza dados para as escalas originais

train_X = train_X.reshape((train_X.shape[0], train_X.shape[2]))
inv_trainPredict = concatenate((trainPredict, train_X[:,1:]), axis=1)
inv_trainPredict = scaler.inverse_transform(inv_trainPredict)
inv_trainPredict = inv_trainPredict[:,0]
inv_testPredict2 = scaler2.inverse_transform(trainPredict)

test_X = test_X.reshape((test_X.shape[0], test_X.shape[2]))
inv_testPredict = concatenate((testPredict, test_X[:,1:]), axis=1)
inv_testPredict = scaler.inverse_transform(inv_testPredict)
inv_testPredict = inv_testPredict[:,0]
inv_trainPredict2 = scaler2.inverse_transform(testPredict)

test_y = test_y.reshape((len(test_y), 1))
inv_y = concatenate((test_y, test_X[:, 1:]), axis=1)
inv_y = scaler.inverse_transform(inv_y)
inv_y = inv_y[:,0]

train_y = train_y.reshape((len(train_y), 1))
inv_yr = concatenate((train_y, train_X[:, 1:]), axis=1)
inv_yr = scaler.inverse_transform(inv_yr)
inv_yr = inv_yr[:,0]

# Calcular RMSE
rmse = sqrt(mean_squared_error(inv_yr, inv_trainPredict))
print('Train RMSE: %.3f' % rmse)
rmse = sqrt(mean_squared_error(inv_y, inv_testPredict))
print('Test RMSE: %.3f' % rmse)
```

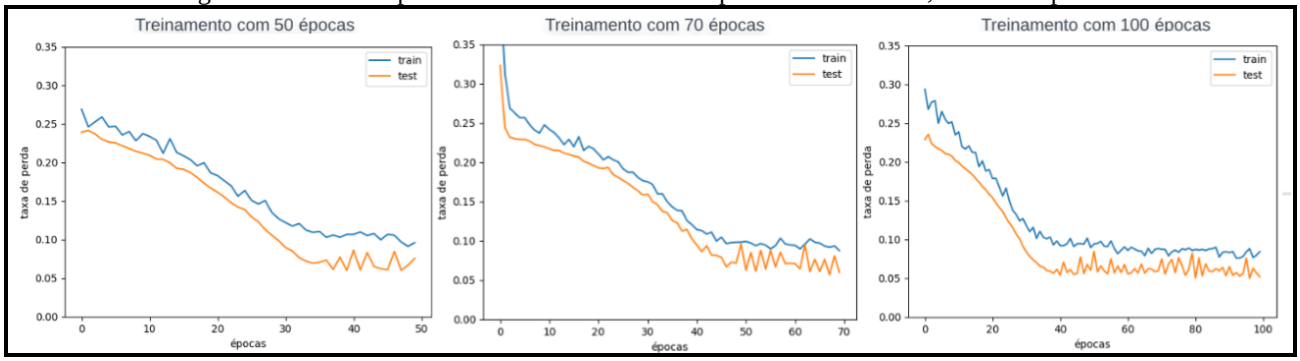
Fonte: elaborado pelo autor.

4 RESULTADOS

Nesta seção é realizada a análise dos resultados obtidos com os testes utilizando a rede neural desenvolvida. Os testes foram realizados alterando os diferentes parâmetros utilizados em seu desenvolvimento. Através do treinamento da rede desenvolvida foi possível obter dados relativos ao desempenho do modelo, onde o indicador utilizado para mensurar o desempenho da rede implementada foi o RMSE.

Os primeiros resultados foram observados variando a quantidade de épocas executadas durante o treinamento, isto é, a quantidade de testes que a rede executa varrendo todas as observações da base de dados. Através dos testes pôde-se observar que, com os dados disponíveis, o ajuste dos pesos se estabilizou por volta de 40 iterações. A figura 11 mostra a taxa de perda do modelo, representada pelo eixo Y, em cada época, representada pelo eixo X. É possível observar que, em todos os modelos, a taxa de perda se manteve estável por volta de 40 iterações sobre toda a base de dados, ou seja, não tendo ajustes significativos nos pesos após essas iterações.

Figura 10 – Taxa de perda do modelo através das épocas utilizando 50, 70 e 100 épocas.

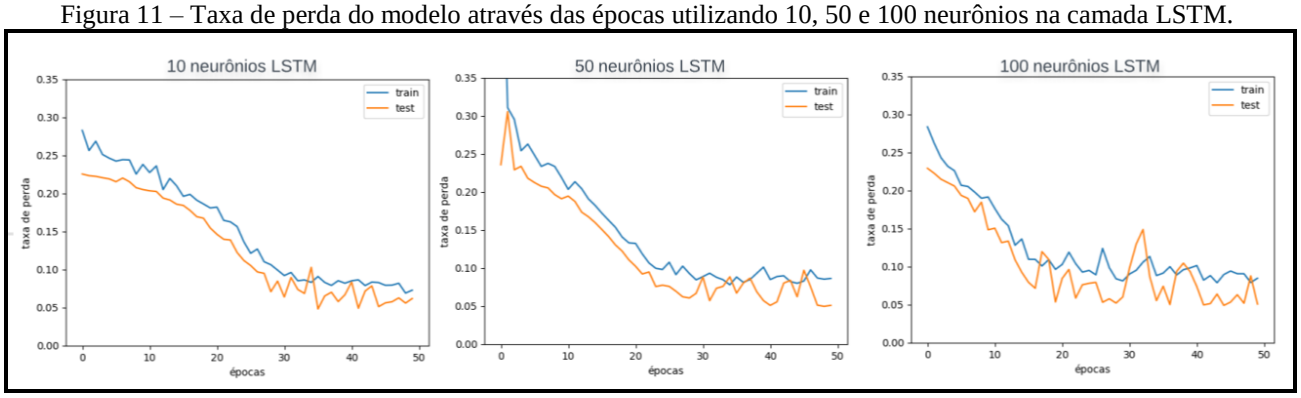


Fonte: elaborado pelo autor.

Também foram realizados testes com a quantidade de neurônios da camada LSTM. A quantidade de neurônios foi definida com base em testes utilizando entre 5 e 100 neurônios nesta camada. Treinamentos executados utilizando mais que 10 neurônios apresentaram uma velocidade de aprendizado maior, porém obtiveram resultados inferiores se

comparado aos testes utilizando 10 camadas. A figura 12 compara os testes realizados utilizando 10, 50 e 100 neurônios. Observa-se que os valores da função de perda, representados pelo eixo Y diminuem durante o treinamento, se estabilizando entre 20 e 30 iterações da rede, demonstrando que, após essa quantidade de iterações, independentemente da quantidade de neurônios, os pesos tendem a não ser mais ajustados, ou seja, a rede já não aprende mais com os dados de treinamento.

Figura 11 – Taxa de perda do modelo através das épocas utilizando 10, 50 e 100 neurônios na camada LSTM.



Fonte: elaborado pelo autor.

A tabela 2 mostra os valores de RMSE resultantes dos testes utilizando 10, 50 e 100 neurônios na camada LSTM. Foram realizados três testes com cada modelo, o valor de RMSE de cada teste representa o erro médio em gramas das predições realizadas pelo modelo. Pode-se observar que a média do resultado dos três treinamentos utilizando 10 neurônios obteve o menor RMSE médio, que foi de 2,629 gramas, enquanto os testes com 50 e 100 neurônios apresentaram médias de 2,676 e 2,736 gramas respectivamente.

Tabela 2 – RMSE dos testes com 10, 50 e 100 neurônios na camada LSTM.

	Teste 1		**Teste 2**		**Teste 3**		
Neurônios na camada LSTM	**RMSE treino**	**RMSE teste**	**RMSE treino**	**RMSE teste**	**RMSE treino**	**RMSE teste**	**Média**
10	2,556	2,673	2,562	2,696	2,585	2,697	**2,629**
50	2,570	2,682	2,670	2,725	2,586	2,690	**2,676**
100	2,825	2,769	2,608	2,773	2,628	2,703	**2,736**

Fonte: elaborado pelo autor.

Além dos testes para definir a quantidade de neurônios, foram realizados testes entre as funções de ativação funções `relu`, `tanh` e `sigmoid` para estabelecer a função de ativação a ser utilizada pela camada LSTM. Estas funções visam controlar os valores computados por cada neurônio para evitar que estes se tornem muito altos. A tabela 3 mostra os valores de RMSE resultantes dos testes utilizando estas funções de ativação. Foram realizados três testes com cada modelo, o valor de RMSE de cada teste representa o erro médio em gramas das predições realizadas pelo modelo. Pode-se observar que a média do resultado dos três treinamentos utilizando a função de ativação `sigmoid` obteve o menor RMSE médio de 2,676 gramas, enquanto as funções `relu` e `tanh` apresentaram RMSEs de 2,697 e 2,764 gramas respectivamente,

Tabela 3 – RMSE dos testes com as funções de ativação `relu`, `tanh` e `sigmoid`

	Teste 1		**Teste 2**		**Teste 3**		
Função de Ativação	**RMSE treino**	**RMSE teste**	**RMSE treino**	**RMSE teste**	**RMSE treino**	**RMSE teste**	**Média**
Relu	2,665	2,77	2,615	2,783	2,62	2,728	**2,697**
tanh	2,757	2,799	2,736	2,786	2,669	2,770	**2,764**
sigmoid	2,57	2,682	2,67	2,725	2,586	2,69	**2,676**

Fonte: elaborado pelo autor.

Assim como na camada LSTM, o critério para definir a quantidade de neurônios na camada intermediária foi a realização de testes com diferentes quantidades (entre 5 e 100) conforme ilustrado na tabela 4, na qual se pode observar os valores de RMSE resultantes dos testes utilizando 5, 10, 50 e 100 neurônios. Foram realizados três testes com cada modelo, onde o valor de RMSE de cada teste representa o erro médio em gramas das predições realizadas pelo modelo. Pode-se observar que a média do resultado dos três treinamentos utilizando 10 neurônios obteve o menor RMSE médio, de 2,587 gramas, enquanto os treinamentos utilizando 5, 50 e 100 neurônios apresentaram RMSEs médios de 2,619, 2,659 e 2,667 gramas, respectivamente.

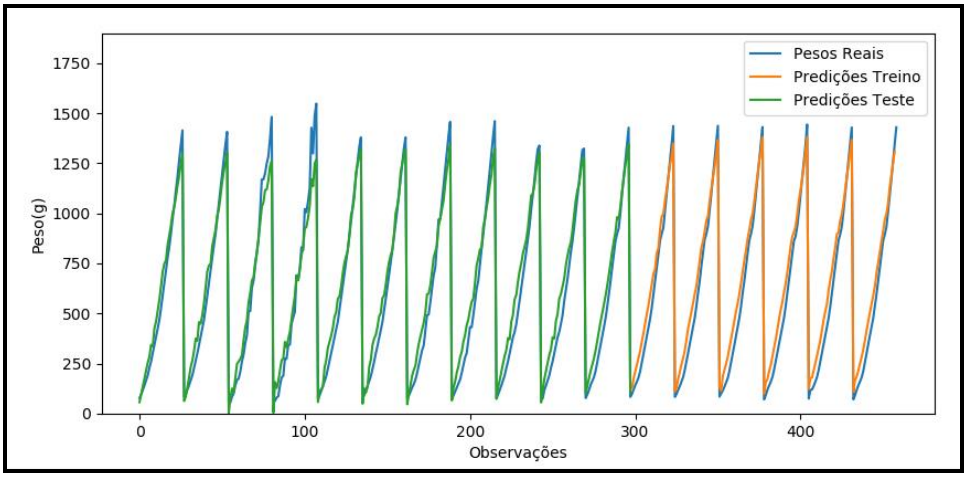
Tabela 4 - RMSE dos testes com 10, 50 e 100 neurônios na camada intermediária.

	Teste 1		Teste 2		Teste 3		
Neurônios segunda camada	**RMSE Treino**	**RMSE Teste**	**RMSE Treino**	**RMSE Teste**	**RMSE Treino**	**RMSE Teste**	**Média**
5	2,548	2,633	2,601	2,626	2,611	2,692	**2,619**
10	2,608	2,732	2,565	2,677	2,531	2,369	**2,587**
50	2,584	2,679	2,594	2,736	2,638	2,710	**2,659**
100	2,586	2,686	2,586	2,737	2,647	2,687	**2,667**

Fonte: elaborado pelo autor.

Com o modelo final desenvolvido, foram realizadas predições utilizando os dados de treino e testes. A figura 13 apresenta um comparativo entre os valores reais e os preditos pela rede, no qual é possível visualizar o peso real das aves para cada exemplo da base de dados (linha azul) e as predições da rede para cada peso durante o treinamento da rede com os dados de treinamento e de teste, em verde e laranja, respectivamente. A sobreposição das linhas de Pesos Reais e Predições mostra que a rede foi capaz de predizer a curva de peso das aves durante a maior parte dos dias durante o seu ciclo de vida, principalmente na fase de testes. Entretanto, pôde-se observar que ao tentar predizer o peso nos últimos dias de vida (os pontos mais altos das linhas), a rede não alcançou a mesma precisão.

Figura 12 – Comparativo entre os dados reais de peso das aves e as predições durante as fases de treino e teste da rede



Fonte: elaborado pelo autor.

O modelo final apresentou os melhores resultados de RMSE durante o treinamento, chegando a 2,619 gramas de peso de variação com os dados de treino e 2,413 gramas com os dados de teste, resultando em um RMSE médio de 2,516 gramas de peso. A tabela 5 faz uma comparação com trabalho correlato 3, de Ferraz et al. (2014), sendo possível observar que modelo LSTM apresentou um valor de RMSE médio 14,114 gramas menor que a rede neural convencional desenvolvida pelo autor. A tabela também mostra a quantidade de dados disponibilizados para o treinamento, e a quantidade de neurônios utilizados nas camadas. Os trabalhos correlatos de Carvalho (2017) e Faridi et al. (2014) não foram incluídos na tabela de comparação, pois, por realizarem a predições em contextos diferentes (respectivamente temperatura superficial e peso de ovos) seus RMSEs estão em escalas diferentes ao do trabalho atual.

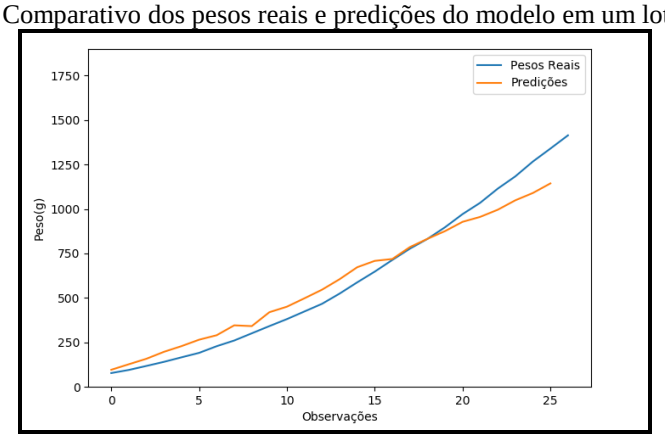
Tabela 5 – Comparação do RMSE resultante do modelo final desenvolvido com o modelo de Ferraz et al (2014)

Trabalho	**RMSE treino**	**Numero de observações**	**Camadas/Neurônios utilizados**
Ferraz et al (2014)	16,63	291	100
Protótipo atual	2,619	495	20

Fonte: elaborado pelo autor.

Entretanto, ao comparar os dados reais e preditos de um lote individual, pode-se notar diferenças superiores ao cálculo médio do RMSE. A figura 14 mostra a comparação entre os pesos reais e pesos preditos de um lote individual. Na figura é possível observar que as predições do modelo (em laranja) tiveram variação positiva (acima do peso real) entre o primeiro e próximo ao décimo-sexto dia de vida, quando passou a ter variação negativa em suas predições (abaixo do peso real), mostrando que o modelo teve uma precisão menor ao testar lotes individualmente.

Figura 13 – Comparativo dos pesos reais e predições do modelo em um lote individual



Fonte: elaborado pelo autor.

A tabela 6 apresenta os valores dos dados reais e preditos de um lote individual. A tabela mostra a idade, temperatura (em Cº), Umidade Relativa (em UR%) e peso real (em gramas), o peso predito pela rede a partir do segundo dia, visto que a predição sempre ocorre para o dia seguinte, a diferença dos valores (peso predito menos o peso real) e o percentual que a diferença representa em relação ao peso no dia. É possível perceber diferenças (peso predito menos o peso real) entre 0,94 gramas e -270,40 gramas. Na maioria das predições individuais a diferença foi superior aos resultados calculados pela rede durante o seu treinamento, de 2,516 gramas. As diferenças entre os resultados reais e preditos nos testes individuais resultaram em um erro percentual médio de 10,099 %.

Tabela 6 – Comparação entre peso real e predito pela rede

Idade	Temperatura (C°)	UR(%)	PESO (g)	PESO PREDITO (g)	DIFERENÇA (g)	DIFERENÇA (%)
1	33,81	70	77,50	-	-	-
2	33,16	75	95,00	95,94	0,94	0,989
3	32,79	78	117,50	127,02	9,52	8,102
4	32,05	78	140,50	157,81	17,31	12,320
5	31,6	82	166,00	196,96	30,96	18,651
6	31,14	82	191,00	228,94	37,94	19,864
7	31,69	83	228,50	265,02	36,52	15,982
8	30,06	82	260,50	290,21	29,71	11,405
9	32,97	81	301,00	345,90	44,90	14,917
10	29,79	82	341,50	341,54	0,04	0,012
11	30,05	82	381,00	419,60	38,60	10,131
12	29,16	83	424,00	450,77	26,77	6,314
13	28,18	84	467,00	498,01	31,01	6,640
14	26,85	84	524,50	546,47	21,97	4,189
15	25,18	82	587,50	605,42	17,92	3,050
16	25,62	82	648,00	672,59	24,59	3,795
17	27,85	85	713,50	708,43	-5,07	-0,711
18	26,06	85	777,00	719,00	-58,00	-7,465
19	25,79	82	832,00	785,96	-46,04	-5,534
20	25,79	82	897,50	832,39	-65,11	-7,255
21	25,27	83	971,50	875,84	-95,66	-9,847
22	26,41	83	1035,00	928,00	-107,00	-10,338
23	27,01	83	1114,50	955,76	-158,74	-14,243
24	26,42	82	1183,50	995,90	-187,60	-15,851
25	26,97	83	1267,00	1048,92	-218,08	-17,212
26	26,32	83	1340,00	1090,12	-249,88	-18,648
27	26,5	83	1414,50	1144,10	-270,40	-19,116
					Dif. média(g):	**40,48**
					% Dif. média:	**10,099**

Fonte: elaborado pelo autor.

Os resultados apresentados mostram que o protótipo foi capaz de prever com precisão o peso das aves com os dados disponíveis durante o seu treinamento, porém, também mostram que não se alcançou a mesma precisão ao predizer o peso das aves utilizando um lote individual. Este resultado sugere que o modelo tenha passado pelo problema de *overfitting* durante o seu treinamento, ou seja, ter se adequado 'demais' à base de dados, tendo predições precisas durante o treino, porém perdendo a precisão ao realizar novas predições. Não foi possível identificar a causa exata da disparidade entre os resultados de treinamento e com lotes individuais. Entretanto, se sugere que a quantidade reduzida de dados utilizadas para os treinamentos possa ter impactado nos resultados finais e na sua disparidade.

5 CONCLUSÕES

Este trabalho apresentou o desenvolvimento de um protótipo para predição de peso de aves utilizando o modelo de rede neural recorrente LSTM. O protótipo desenvolvido foi capaz de predizer o peso das aves considerando os fatores ambienteis aos quais estavam submetidas, com uma taxa de RMSE de 2,516 gramas durante os treinos. Como limitações, o modelo teve seus melhores resultados ao predizer o peso nas primeiras semanas de vida das aves, porém não atingiu a mesma precisão ao predizer o peso nos últimos dias de vida. Além disso, predições realizadas com lotes individuais também apresentaram precisão inferior à observada durante o treinamento, onde se obteve uma taxa média satisfatória de 40,48 gramas. Entende-se que essa perda da precisão é resultante da insuficiência de dados disponíveis para o treinamento da rede, o que pode ter resultado no *overfitting* do modelo.

Por este motivo, como extensões para este trabalho seria essencial treinar o protótipo desenvolvido utilizando uma base de dados maior, visto que a base disponível para os testes possui apenas 459 observações. Além disso, seria necessário o estudo de outros métodos e/ou modelos como Redes Neurais Recorrentes mais simples, ou variações da LSTM, como o modelo *peepholes* conections que, a cada execução, adiciona informações adicionais do estado para as funções de ativação `sigmoid`, ou a variação *Gated recurrent Unit* que reúne o *input gate* layer e a camada `tanh` das células LSTM em uma única unidade de processamento. Tais modelos poderiam ser treinados para buscar melhores resultados ao predizer o peso das aves ao utilizar lotes individuais, assim como melhorar as predições de seus últimos dias de vida, de forma a possibilitar a previsão da data em que o lote poderá alcançar o peso desejado para o abate, otimizando os recursos da cadeia produtiva de frango de corte.

Dessa forma, os resultados confirmam que o protótipo tem potencial para ser utilizado como ferramenta para a simulação de cenários de microclima alternativos, a fim de buscar melhores condições ambientais na criação de aves. Com isso seria possível propiciar um melhor desempenho no seu ganho de peso, bem como ser integrado a sistemas de ambiência de granjas para controle automático do microclima das aves.

REFERÊNCIAS

AMARAL, A. G. et al. Efeito do ambiente de produção sobre frangos de corte sexados criados em galpão comercial. **Arquivo Brasileiro de Medicina Veterinária e Zootecnia**, Lavras, MG, v. 63, n. 3, p.649-658, jan. 2011.

ASSOCIAÇÃO BRASILEIRA DE PROTEÍNA ANIMAL. **Resumo do Setor de Aves.** 2018. Disponível em: <http://abpa-br.com.br/setores/avicultura/resumo>. Acesso em: 31 mar. 2018.

AVIAGEN. **ROSS Envioronmental Managament in the Broiler House.** 2010. Disponível em: <http://pt.aviagen.com/assets/Tech_Center/Ross_Broiler/Ross_Environmental_Management_in_the_Broiler_House.pdf>. Acesso em: 31 out. 2018.

BORGES, S. A.; MAIORKA, A.; SILVA, A. V. F. Heat stress physiology and eletrolites for broilers. **Ciência Rural**, v. 33, n. 5, p. 975-981, 2003

CARVALHO, K. A. **Inteligência Artificial Aplicada à Predição da Temperatura Superficial de Frangos de Corte.** 2017. 47 f. Dissertação (Mestrado) - Curso de Engenharia de Sistemas e Automação, Universidade Federal de Lavras, Lavras, Mg, 2017.

CARVALHO-CURI, T. M. R. de; MOURA, D. J. **Ambiência de precisão na avicultura de corte: sua importância e ferramentas.** 2017. Disponível em: <https://www.aviculturaindustrial.com.br/imprensa/ambiencia-de-precisao-na-avicultura-de-corte-sua-importancia-e-ferramentas/20160722-092338-y060>. Acesso em: 31 mar. 2018.

DOZIER III, W. A.; PURSWELL, J. L.; BRANTON, S. L. Growth responses of male broilers subjected to high air velocity for either twelve or twenty-four hours from thirty-seven to fifty-one days of age. **Journal of Applied Poultry Research**, v. 15, n. 3, p. 362-366, 2006.

EMPRESA BRASILEIRA DE PESQUISA AGROPECUÁRIA. **Estatísticas de frango de corte no mundo.** 2018. Disponível em: <https://www.embrapa.br/suinos-e-aves/cias/estatisticas/frangos/mundo>. Acesso em: 31 mar. 2018.

FARIDI, A.; FRANCE, J.; GOLIAN, A. Neural network models for predicting early egg weight in broiler breeder hens. **Journal of Applied Poultry Research**, v. 22, n. 1, p. 1-8, 2013.

FAUSETT, L. **Fundamentals of neural networks:** architectures, algorithms and applications. Upper Saddle River: Prentice-hall, Inc., 1994. 480 p.

FERRAZ, P. F. P. et al. Predicting chick body mass by artificial intelligence-based models. **Pesquisa Agropecuária Brasileira**, v. 49, n. 7, p. 559-568, 2014.

GOMES, D. T. **Uso de redes neurais recorrentes para previsão de séries temporais financeiras.** 2005. 138 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Instituto de Matemática, Estatística e Computação Científica, Universidade Federal de Minas Gerais, Belo Horizonte, Mg, 2005.

GRAVES, A. **Supervised Sequence Labelling with Recurrent Neural Networks.** 2008. 114 f. Tese (Doutorado) - Curso de Ciência da Computação, Universidade Técnica de Munique, Munique, 2008.

HAYKIN, Simon. **Redes Neurais: Princípios e Prática.** 2. ed. Porto Alegre, Rs: Bookman, 2003. 898 p.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural computation**, v. 9, n. 8, p. 1735-1780, 1997.

KERAS. **The Python Deep Learning library.** 2015. Disponível em: <https://keras.io/>. Acesso em: 01 jun. 2018.

KOVÁCS, Z. L. **Redes neurais:** fundamentos e aplicações. 4. ed. São Paulo, Sp: Livraria da Física, 2006. 174 p.

LECUN, Y. et al. **Efficient BackProp:** Neural Networks: Tricks of the trade. Berlin: Springer, 2012. 44 p.

LU, Q.; WEN, J.; ZHANG, H. Effect of chronic heat exposure on fat deposition and meat quality in two genetic types of chicken. **Poultry Science**, v. 86, n. 6, p. 1059-1064, 2007.

LUCIDCHART. **Lucidchart.** 2008. Disponível em: <https://www.lucidchart.com/>. Acesso em: 25 nov. 2018.

MEDSKER, L.; JAIN, L. **Recurrent Neural Networks:** Design and Applications. Boca Raton: Crc Press, Inc., 2001. 389 p.

MENEGALI, I. et al. Comportamento de variáveis climáticas em sistemas de ventilação mínima para produção de pintos de corte. **Revista Brasileira de Engenharia Agrícola e Ambiental [online]**, Campina Grande, v. 17, n. 1, p.106-113, jan. 2013.

COLAH, C. **Understanding LSTM Networks.** 2015. Disponível em: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>. Acesso em: 17 nov. 2018.

OLIVEIRA, D. R. M. S. et al. **Issues of sustainability on the Brazilian broiler meat production chain**. Proceedings of the International Conference on Advances in Production Management Systems; 2012; Rhodes. Philadelphia: IFIP; 2012

PEREIRA, D. F.; NÄÄS, I. A. Estimativa do conforto de matrizes de frango de corte baseada em análise do comportamento de preferência térmica. **Engenharia Agrícola**, p. 315-321, 2005.

TORRETA, M. **Fatores que afetam a conversão alimentar em frangos de corte.** 2017. Disponível em: <http://www.agroceresmultimix.com.br/blog/fatores-que-afetam-conversao-alimentar-em-frangos-de-corte/>. Acesso em: 25 nov. 2018.

TWOMEY, J. M.; SMITH, A. E.. **Performance Measures, Consistency, and Power for Artificial Neural Network Models.** 21. ed. Oxford, New York: Pergamon Press, 1995. 16 p.